Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering

Leon Nicola Cinquemani

NationSim

A story-driven approach to Agent-Based Modeling of Nations interacting.

Master's thesis in Applied Computer Science
Supervisor: Christopher Frantz
January 2023

NTNU
Norwegian University of
Science and Technology

Leon Nicola Cinquemani

NationSim

A story-driven approach to Agent-Based Modeling of Nations interacting.

Master's thesis in Applied Computer Science
Supervisor: Christopher Frantz
January 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering

NationSim, a story-driven approach to Agent-Based Modeling of Nations interacting.

Leon Nicola Cinquemani

Abstract

Agent-based simulation has been well integrated into the field of social sciences, with social simulations being in use both in the field of science, as well as at a government level for some time now.
This thesis details the process of creating a framework that sets out to simulate a variety of nations and their interactions, to gain insights into what rules guide these interactions, and what effects they have on both the nations involved, as well as those that are bystanders in the matter.
It then also validates the data produced by this framework against observable trends in the economies and development of real nations.

The thesis also discusses the application of two separate frameworks as part of the creation process of the product, where one light-weight framework is used for the rapid prototyping of ideas, and one more complex and object-oriented framework is used to create an orderly, low coupling enforcing implementation of these ideas into the finished program.
Over the course of the thesis, it discusses both the benefits and the drawbacks of utilizing this workflow and aims to prove that for a significant subset of projects, this is a worthwhile investment of time.

Sammendrag

Agent-based simulasjon har lenge vært en viktig del av samfunnsvitenskap, både innenfor vitenskap og innenfor regjeringsarbeid.
Denne Avhandlingen dekker arbeidet gjort for å lage et rammeverk som kan simulere en rekke nasjoner og deres handlinger på både dem selv og en annen, for å få en oversikt over hva det er som driver disse handlingene, og hvilken effekter de har, både på aktørene og på bistående.
Avhandling så også sjekker data-en produsert av rammeverket mot tydelige trender innen økonomiene og utvikling av reelle land.

I tillegg diskuterer avhandlingen en arbeidsplan for bruken av to rammeverk til å lage agent-baserte modeller, der et lettvekts rammeverk blir brukt til prototyping av ideer, mens et mer kompleks, objektorientert rammeverk blir brukt til å lage en oversiktlig implementasjon av disse prototypene, med lave koblinger mellom delene av systemet.
I løpet av avhandlingen blir det vist til både fordeler og ulemper med denne måten å arbeide på, og en påstand er gjort at den er til fordel for en vid rekke prosjekter.

Contents

Figures

Tables

Code Listings

Acronyms

ABM Agent Based Modeling. 4, 7, 8

ACE Agent-based Computational Economics. 8

Chapter 1

Introduction

1.1 Background

1.1.1 My belief in stories

As humanity's exponential consumption of media only grows, and the continued technology able to provide this media in vast quantities and vastly unique ways only becomes more advanced, the need for novel entertainment increases on a near-yearly basis.

This necessitates the continued creation of novel entertainment media, ranging across all forms of art and entertainment. New music, new books, new art. New movies, new games, new theater. Humanity's hunger for stories, sparked alongside the first-ever campfire, has only continued to grow since the first tales were told over the crackling of the firewood.

From tales told around campfires to images used to accentuate verbal tales, the very concept of religion is funded in the art of telling stories to make sense of the world. Stories ranging from fabricated tales to the concrete retelling of events that happened in the past. It is no wonder then that the Latin word for history, *historia* can mean both "recorded knowledge of past events", but also "story" and "narrative".

In a way, to this day everything humanity "knows" about itself and the universe is a story.
By the inherent uncertainty in the universe and the seeming impossibility of understanding its concrete rules, if, indeed, any rule concerning the entire vastness of existence can truly be "*concrete*" across all cases, our attempts at making sense of both it and the mechanisms that guide it, are typically framed as "hypotheses" and "theories".

In a very real sense then, we must come to terms with the fact that a *theory* is

not a *certainty*.
By acknowledging the fact that the entire universe is thus viewed by modern scientists through a set of theorems that do not constitute concrete rules, we have to conclude that science too, in the widest sense, follows in the footsteps of storytelling.
In a way, even with all the enlightenment of modern sciences, we are still stuck in Plato's cave, where the things we perceive as concrete realities are at best well informed stories.

Likewise, the very concept of any recorded history requires the history to first be recorded, undoubtedly by a human.
This means by the nature of human communication, even in a scientific background, that recorded historical events like all writing *must* be tainted by the biases of the recorder.

Just as it is impossible to separate the art from the artist, so too is it impossible to separate the writing from its writer.
So long as history is created not by an entirely impartial bystander with no emotional investment in the matter, nor personal biases, it will always result in a biased product.

This is obvious in major ancient historical records, from the fall of troy to the religious history of the Vikings, as recorded by Snorre Sturlason. Both the historical inaccuracies of the Illiad, as well as the likely Christianization of the original Norse-pagan beliefs under the Christian Sturlason showcase the nature of history, namely that it molds to the people that end up telling others about it.

With such a major portion of human life seemingly centered around the consumption and production of stories, it is unsurprising that the market for these concepts is quite vast, and the amount of, indeed, consumption of stories is thus understandably massive.
Humanity's ability to strive towards ever greater skill in delivering a story that evokes emotion, provokes thought, or just gets across our viewpoint on a matter, has left us with a vast, and varied array of tales, across various genres.

Thus, there are many stories to be told and almost as many ways to tell a story.
And one recent way of telling stories is simulation.
While some may cry out at describing a highly advanced tool for modeling the interactions between physical and/or societal phenomena as "storytelling", in a very real sense, that is exactly what simulations are.
No matter how broad the creator's understanding of the subject may be, in the end, there is no way to model all possibilities and variables that are inherent in every subject, from the life of single-celled organisms to the fall of a single grain of dust through the atmosphere.

Simulation relies on taking the unfathomable vastness of reality and squeezing it into a box that the human mind can comprehend and find a feeling of certainty in.

No simulation can ever predict the future accurately, for to do so all important variables to every possible event, let alone their interaction, would have to be known.
But that is not the point of a story, not even that of history, I would pose.
The meaning behind a story, behind all stories is to communicate something to the audience, something they had not before, be that understanding, a certain feeling on a matter, or even just the satisfaction of engaging with something interesting.
Thus I believe is the purpose of all stories, from fireside tales, fairy tales, and best-selling novels, to historical accounts, research papers, and biographies.
In the end, it is to get across a message, to showcase a tiny part of existence that the person telling the story thinks is important.

Does the tale of Achilles and Patroclus have any direct impact on the life of those that read the Illiad today?
Maybe.
Maybe the butterfly effect means that somewhere during the real events that are the base for the story, something happened that is affecting the life of the person reading the tale right now.
But that is not the point of the story, is it?
The point is that the lessons learned in the past, the core understanding that Homer wished to get across to his audience, is something that can inspire people and change the world even still.
And just the same, the point of a simulation is not to provide a perfect understanding of the world, a flawless telling of what the future, past, hypothetical event, or minute interaction, is like, but to instead transmit some information that the creator believes is important.

As we lack certainty in our understanding of the rules that govern the world, it is the duty of the scientist to come up with, yes, stories.
Stories that can explain the phenomena that are observed in the real world.
From the earliest idea that there had to be a point at which an item could not longer be divided into smaller pieces, because the smallest possible piece had been reached, to the discovery of atoms, from Newtonian physics to Relativity, science relies on somebody making up a story about how the world works and then testing to see if it holds up.
If there is nothing that contradicts the story, and more and more people believe in the story, it becomes a cornerstone of science, until the point where something is found that contradicts it, and a new story is told that explains that new observation as well.

From this belief, this project has been born.

The firm belief that observing the world and finding stories to tell is a core part of the human experience in all its forms.
A simulation, in the end, is a simpler world for us to look at, one which might show us interactions and hint at patterns that have not previously been observed in the real world due to all the noise of interfering variables.
And from that stories might be told, stories that have value in the chaos of the real world.

1.1.2 My story

The story this thesis is meant to tell is one that is becoming increasingly relevant in today's world.
Where nations and peoples are ravaged by a pandemic that has destroyed both lives and lively-hoods, where war rages on lands that had hoped they would be at peace for many more generations, where the climate, energy, and market are in a crisis.
A world that now more than ever is changing due to the actions and interactions of nations.
The story this thesis is meant to tell is one of Nations cooperating, trading, fighting, changing, living, and dying, a story that may carry some insights that can apply to the real world.

And where the caveman used his voice and a bonfire, the renaissance painter used his art, and the modern writer uses his literary works, this thesis uses Agent Based Modeling (ABM).
Because a good simulation, or more precisely, a good framework for simulations, can tell a thousand stories.
Whichever story the user wishes to tell, provided they know how to use the framework.
Over the course of nearly a year and a half, I have worked on the creation of a framework that can tell the story I wish to tell.
That can simulate many, many nations interacting with one another, and is robust enough to let my audience explore many potential stories by having a say in *how* those nations interact.

This thesis too is a story.
It does not contain absolute truth, because inevitably, it will be altered by my biases, and the lens through which it describes the long, arduous process of creating a functional simulation framework, is unique to the author as well.
But hopefully, by the end of this thesis, the most important parts of the process of the creation of NationSim, and its capabilities will have been well explained.

1.1.3 Planned Contributions

Originally the goal of the project was to make a tool to assist writers in creating interesting historical data for fictional settings.

As a hobbyist writer I, at the time of starting the project, found myself with the issue of requiring a lot of historical data for the nations of the fictional worlds I was working on, as I wanted to make them feel like well-established members of an international community, with a history of conflicts and cooperation to look back on.
The vast quantity of fictional works that are being produced annually emboldened me in the pursuit of a way to create this data without needing to spend many hours combing through real-world historical data every time, as it seemed as though such a product would certainly find use within that community. It seemed to me that such a product would enable authors to generate historical data within moments, and use that as scaffolding upon which they could build the history of their worlds nations.

As such, the original idea for the project was to create a simulation of nations interacting and trying to survive throughout a predetermined period of history, and then gather a list of important historical events such as wars fought, alliances formed, and trade deals struck for each nation.
Over the course of the project, however, another use for NationSim came into the spotlight.
The ability to simulate nations and their interactions is highly topical at the current time, as the war between Ukraine and Russia is still in full swing, and the coronavirus still heavily impacts the international economy.

As the simulation of nation interactions has taken precedence over the simulation of historical events, the focus of the project has shifted from presenting the user with a set of historical events, to instead showing the user *why* nations end up in those situations.
The main thing that "NationSim" seeks to provide is the ability for its users to look at nations and their interactions and draw conclusions about the underlying mechanisms that decide how they end up interacting, as well as what the contributing factors to if they end up prosperous or not are.

1.2 Literature Review

The idea of using simulations to gain insights into society is far from a new one, with scientists attempting to use computers to aid in understanding and even predicting social phenomena since the early 1960s (N. Gilbert and K. Troitzsch, *Simulation for the social scientist*. McGraw-Hill Education (UK), 2005[1]), and the

concept of using simulation to model nations is just as old.
Not long after the first attempts at modeling society via computer simulations, the 1970s saw the rise of large-scale world models, like WORLD II (Forrester [2]), which attempted to model complex interactions between large, world-spanning phenomena like pollution, population, and capital gain.

This attempt however was doomed to be a difficult undertaking, as attempting to model such a massive set of interactions and create a predictive model means that untold numbers of variables need to be taken into consideration.
And with a lack of accurate knowledge of each of these parameters, the simulation makes increasingly unrealistic predictions, which can be showcased by the fact that if the model was used to predict birthrates going backward into the 1800s, it predicted that the world population sank from 6 billion to 1.7 billion between 1880 and 1900, which thankfully is not the case.

Originally, social scientists were focused on using discrete event simulations and simulations based on system dynamics (Gilbert and Troitzsch [1]). The former was used in order to analyze waiting times in queues or time taken for emergency services to reach a destination (Kolesar *et al.* [3]), while the latter was used in predictive work for example the world economy (Meadows *et al.* [4]).
At the time, strong predictions were made but ended up being criticized for their low reliability and lack of evidence backing the assumptions made with the model's parameters.

After that came Microsimulations (Orcutt *et al.* [5]), which observed a large set of simulated subjects, such as individuals, households, or firms.
Then for every timestep of the simulation, a set of transformative functions is run on each subject, changing their state with a percentage chance.
After Microsimulations the field remains relatively quiet until the 1990s when the first developments are made in the field of using multi-agent models to model social interactions([1]).
These include early attempts at utilizing cellular automata, grids of "cells", each of which has a state that is influenced by the states of the cells surrounding it(Toffoli and Margolus [6]), and advancements in the fields of artificial intelligence allowing the "agents" of the model to reason and even learn over the course of the simulation(Doran [7]).

From mathematical models to agent-based models, simulation has become more and more of an invaluable tool in the search for understanding social phenomena.
An important factor in understanding society is the concept of *emergence*, the process by which the rules govern the individuals within a group form rules that govern the entire group.
No variable in an agent may linearly be related to the emergent behavior, but the

combination of the agent's behaviors changes how the agents behave as a whole (Simon [8]).
These emergent behaviors are an important part of NationSim, as the behaviors that control whether or not a nation makes a profit from interacting with others via a market, and how the nations impact one another rely on emergent behaviors in the nations. One nation not being able to meet its requirements for a certain resource drives up the price of that resource locally, which results in the price on the open market changing.
Suddenly producing that resource and selling it on the open market creates bigger profits, which causes more nations to do so.
Thus, one nation struggling might create opportunities for other nations to turn a profit.

But a nation is a big thing to model. There is a myriad of constituent parts that make up a nation, indeed it might be correct to say that a nation itself is a concept that is emergent from the interactions of many constituent parts.
This is where ABM's can shine, as they allow the creator to decide to what level of granularity they want to model such emergent concepts.
A crowd can be modeled as individuals with their desires for where to go, or as an amorphous blob moving around based on some more abstract rules.
We can aggregate people in populations, populations into nations, and nations into international societies, and each step of these aggregations can be modeled as agents, down to the desired granularity.
Seen in that light, ABMs are ideal for this use-case, where the level of aggregation desired is entirely up to the programmer.

As for NationSim, I had only a limited amount of time, resources, and accurate knowledge of the inner workings of a nation to work with when it came to modeling the wide-reaching complexities of what systems make up a nation and the rules that govern how they function.
While the ideal goal would of course be modeling everything from population dynamics, internal and external institutions, environmental factors, etc, realistically I only had so much time when working on the project.
As the original goal of the project was to create a writing tool, I decided to focus on three areas I felt were the most necessary for creating a compelling story, that being economy, environmental factors, and warfare. In the end, environmental factors were omitted relatively early on and replaced with a more complex simulation of the internal workings of the government and population growth than originally planned.
This was partially due to time constraints not allowing for a sufficiently complex model of the many factors that go into how the world around a nation behaves, and how the nation and the environment impact one another, but also due to the fact that the environment out of the frameworks presented seemed like the one that was the least likely to aid in the creation of a realistic, fascinating narrative

about the nations being simulated.
While great natural catastrophes impacting nations are an important part of history, they are rare enough that it is not out of the question that they might never come to pass in a simulation, and thus it was decided that if disasters were wanted, there were better ways to incorporate this into the simulation that going through the effort of creating an entire module for it.
In the end, the environment module was dropped as I believed it would have not been worth the time it would have taken to create.

As for the matter of economics, simulation has been used in a variety of contexts around the field of simulating markets, economics in general, customer behavior, planning the economy of major nations (Farmer and Foley [9]), etc. The use of agent-based modeling of economics, the field of Agent-based Computational Economics (ACE) is still fairly young, the groundwork being laid no sooner than 1996 [10].
Since that point, agent-based models have been used to simulate traders in an open market; such as by Chen and Yeh [11] in their model of traders learning forecasting models and taking them into practice, labor market models; such as the one by Tesfatsion [12] which matches employers and employees using game-theory and continuously making workers and employers seek out new opportunities, and prediction of new models for power markets; like the one made by in the 2021 Wiley/IEEE press book[13].
This matches to an extent the use of ABMs in NationSim for the creation of an international open market, alongside the nation's trading on it, and is highly relevant to the project as a result.
ABM has also been useful for creating models of macro-economics(LeBaron and Tesfatsion [14]), which is very much what this thesis concerns itself with as well.

Another important factor in trying to model a nation is attempting to model the population, particularly the way it changes over time.
As nations require people in them in order to function, and economies require labor, ensuring that the nation's population growth function is as close to reality as I can manage to get it is an important part of ensuring the output of the simulation is realistic.
For this, I turned to papers diving into the less-than-obvious and often unclear factors that contribute to the birth and death rates of a population.
Some of these are rather hard to model without going into grand detail about medical factors that would be far out of scope for a simulation about nations, such as genetics and lifestyles (World [15]). Low income however was a prevailing factor(Mackenbach *et al.* [16]) that could be simulated in the model to an extent by looking at the income across the nation and comparing it to the population we can figure out the average income for each member of the population.
Also noteworthy where the findings of the impact that education, healthcare, and urbanization have on the birthrates of nations, as described by (A. Pourreza,

A. Sadeghi, M. Amini-Rarani, R. Khodayari-Zarnaq and H. Jafari, 'Contributing factors to the total fertility rate declining trend in the middle east and north africa: A systemic review,' *Journal of Health, Population and Nutrition*, vol. 40, no. 1, pp. 1–7, 2021).
Inversely, the death rate of a nation could also be modeled with information from the same articles, as particularly (D. World, 'Longevity: Extending life span expectancy.,' 2022, Retrieved January 12, 2023. [Online]. Available: `www.disabled-world.com/fitness/longevity/`) points out a few factors, such as low healthcare, poor lifestyle choices, and of course, overall age of the population, all of which can to an extend be modeled, the lifestyle being possible to somewhat get an approximation for via education levels and income.

Another factor to consider when modeling the actions of nations are actions of other nations, and how they affect the nation in question.
We cannot create a simulation about nations interacting if the actions of nations do not affect other nations, after all.
International relations are a wide field, of course.
Trade, warfare, diplomacy, alliances, and similar concepts are daunting tasks to model in their entirety, but several attempts have been made at gaining an understanding of how and why nations cooperate, go to war, and form empires.

One example of this is Axelrod's work in creating a simulation of nations exerting tribute from each other and forming empires, before eventually collapsing[N. Gilbert, *Emergence in social simulation in gilbert, n. and conte, r.(eds.) artificial societies*, 1995].
The model in question is an example of a cellular automata model, with agents making decisions based on the states of their neighboring agents[6].
In the model, all agents are placed in a ring, with one neighbor on each side.
Every time step a number of agents is chosen to interact with its neighbors, deciding whether to attempt to extort tribute from them or not.
If the neighbor decides not to pay tribute both nations go to war and lose some of their wealth.
For every positive interaction, such as paying tribute, being paid tribute, and joining each other in their wars, nations gain points of commitment against each other, while going to war on opposing sides results in the opposite.
Thus, over time, clusters of closely cooperating actors start to form, and they may then start to collapse at a later point if powerful constituent members lose too much of their wealth in unfavorable wars.
Thus Axelrod's model can model a series of historical events, such as empires collapsing due to smaller constituent nations picking wars that end up dragging the major players into a conflict that drains their wealth, or major conflicts that collapse the most powerful players as they are forced into opposing camps in devastating "world wars".

As warfare, empire building, and the rise and collapse of civilizations is a major part of history and an important factor in how nations act and interact, a module for warfare and diplomacy was originally planned for NationSim.
The sheer complexity of the economy and internal nation modules ended up however forcing the sacrifice of that module due to time constraints, and international relations and warfare have both been relegated to being something to add to the model in the future.

1.3 Research Questions

As we see, there is a lot of work done in modeling economics, international interactions, and a lot of data for studying the growth factors of a nation's population, as well as models that are meant to help predict the course of nations over in the future.

One might ask what the point of walking such a well-trodden path is, and my reasoning for this is two-fold.
Firstly, the fact that a lot of the questions posed in this thesis and a lot of the difficulties tackled have been discussed before shows that these are problems that have historically been viewed as relevant and important.
Secondly, the fact that all of these papers either focus only on parts of what my implementation focuses on, but go into more detail, or they focus on a bigger picture with a less focused view.
Thus "NationSim" slots into a middle point where it is not just a market simulation, and not just a population growth simulation, but rather a combination of many variables that affect both.

As such, I would like to pose and subsequently answer the following questions in my thesis, in order to prove that NationSim accomplished what it set out to do.

- Process:
    1. "R1: Which ABM-frameworks can best support the modeling of complex social systems?"
    2. "R2: What are the benefits and drawbacks of using separate frameworks for prototyping and implementing a complex model?"
- Evaluation:
    1. "R3: What are the consequences on the international community, economically and demographically, when we remove a main producer for an important resource, like Russia, from the international market?"
    2. "R4: When do nations benefit from international trade, and when is it more efficient to remain in autarky?"

1.3.1 Scenarios

Two scenarios where developed with the intent of gaining answers to the research questions posed.
These were specifically designed in order to simulate situations which would force certain interactions between the modeled nations, in order to gain insight into the mechanisms that control the effects these interactions have on the nations involved in them.
In order to gain answers to the Research questions posed, the following two scenarios where created:

1. "What happens to the international economy and the economic growth of other nations if the main producer of a highly desired product disappears from the market?"
2. "What effects does trade between a wealthy, powerful nation and a poor, weak nation have for the two nations economies and prosperity?"

Each scenario was chosen to gain insight into one of the two main research questions, by letting us analyze the variables that make up the nations involved in each scenario.

1.4 Requirements and Risks

1.4.1 User Stories

The first step in planning the actual architecture of the model, now that I knew what I wanted it to do and what I wanted this thesis to be about, was to create a set of requirements that the end product had to fulfil in order for me to consider it a success.
These where originally chosen via a set of user stories as seen in Table 1.1, which have later been made obsolete by the fact that the project changed direction from a storytelling tool to a tool for exploring the actions of nations.

1.4.2 Requirements

Still, despite the fact that most of the user stories by this point are no longer relevant for the framework, the requirements that I synthesised from them are still largely relevant, with lots of them being useful for the validation of our final product later down the line in Chapter 2. See Table 1.2 and Table 1.3 for the full requirements.

1.4.3 Risks

Likewise, the original risks also remained highly relevant, with several of them becoming a reality over the course of the project.
The project shifting gears several times over the course of the year-long implementation process mean that several risks now have incorrect threat assessments, as the shifting focus for the output of the product and the changing of which framework to use for implementation throughout the project ended up altering which risks where the most problematic, and which requirements where realistically feasible.
The original risks can be seen in Table 1.4

Table 1.1: The Original User stories of the project

Label	User type	Description
US:01	Author	As an author, this user wants to be able to run the tool to create a full history of a set of nations to use as scaffolding for his next novel setting.
US:02	Author	As an author, this user wants to be able to fine-tune the simulation to run an exact number of years equal to the age of his setting.
US:03	Author	As an author, this user wants to be able to select the names of the nations in the simulation to match those in his setting to make them easier to differentiate at a glance when looking over results.
US:04	Author	As an author, this user wants to be able to separate nations that reasonably should not be able to get into contact with one another during the simulation.
US:05	Author	As an author, this user wants to be able to store the results of the simulation for extended periods to be able to divide the time he works on them over a large time frame.
US:06	Author	As an author, this user wants to be able to influence the simulation to change the behavior and starting parameters of each nation to represent the different races and cultures of his setting.
US:07	Videogame developer	As a game developer, this user wants to be able to run the simulation in a reasonably short time, to meet his deadlines.
US:08	Videogame developer	As a game developer, this user wants to be able to have some control over how the different nations develop during the simulation to facilitate the story of the game.
US:09	TTRPG player	As a Table-top RPG player, this user wants to be able to alter the setting later down the line to facilitate a new idea, or to incorporate some input from his fellow players.
US:10	TTRPG player	As a Table-top RPG player, this user wants to be able to break the results from the simulation down into small chunks to be developed further as needed.
US:11	Historian	As a hobbyist-historian, this user wants to be able to import historical data for real nations into the framework to create what-if scenarios.

Table 1.2: The Functional Requirements

Label	Related User story	Name	description	importance
Req:01	US:01 & US:02	Runtime	The program must run a full simulation of a set number of years	High
Req:02	US:01 & US:03 & US:06 & US:11	Realism	The simulation must run realistic interactions between a set of predefined nations	High
Req:03	US:03 & US:04 & US:06 & US:11	Adaptable Parameters	The Nations starting parameters need to be able to be influenced by the user	Med-High
Req:04	US:09 & Us:11	File loading	The Simulations parameters need to be able to be populated from pre-existing data	Med
Req:05	US:01 & US:05 & US:10 & US:11	Output Readability	The program needs to output it's results in a format that are easily human readable	High
Req:06	US:09	Output reuse	The program needs to output it's results in a way that allows it to be fed back into the simulation as starting parameters	Med-Low
Req:07	US:01 & US:03 & US:05 & US:10 & US:11	Auditing	The user has to be able to draw conclusions from the output about the state of the individual nations during any point of the Simulation	High
Req:08	US:09	Storeable state	The user has to be able to store the current state of the simulation, in order to be able to continue from it at a later point in time	Med
Req:09	US:08 & US:09	Active control	The user has to be able to pause the simulation during runtime, and to alter the parameters of nations while the simulation is running	Med-Low
Req:10	US:04 & US:06 & US:08 & US:11	Nation behaviour	The user has to be able to set limitations on how nations may interact to model various complex nation relationships	Med-High
Req:11	US:01 & US:09 & US:11	Deterministic RNG	The user needs to be able to rerun the simulation with the same seed and get the same results.	High-Med

Table 1.3: The non-functional Requirements

Label	Related User story	Name	description	importance
Req:12	US:01	Time Units	The simulation needs to run at a predefined timescale of 1 tick = 1 year, updating once per unit of time for each agent, and storing new information about the agents state, so it can later be presented	High
Req:13	US:07	Runtime	The program needs to take into consideration the runtime of the simulation, and ensure it does not become to long. The simulation should take no more than 1 minute for 1000 years to be simulated for 10 nations. This would mean that 1 year for 1 nation should take no more than 6 milliseconds.	Medium
Req:14	US:01 & US:03 & US:05 & US:10	Output format	The Program needs to produce output that is human and machine readable, and user friendly, producing both raw data, graphs, and text based output for the user	High
Req:15	US:01	User friendliness	The program needs to be intuitive to use, taking less than 2 hours to learn while unguided, and less than 1 hour while guided for a majority of users	Medium
Req:16	US:01 & US:07	Reliability	The program needs to be stable during runtime for typical usecases, not crashing outside of outside circumstances or extreme strain due to excessively taxing user input far beyond the expected norm	High
Req:17	US:01 & US:02 & US:03 & US:04 & US:06 & US:08 & US:09 & US:11	Initialization	The Program needs to allow users to easily input custom parameters, either via loading a file, or via randomization with or without a predetermined seed	High-Medium

Table 1.4: Risks and Related requirements

Label	Related Requirements	Description	Likelihood	Severity
Risk:01	All	I might not be able to complete the project in the timeframe given	Med	High
Risk:02	Req:02	The Nations decision making is not realistic	High	High
Risk:03	Req:05 & Req:07	The output of the simulation is not user readable	Medium	High
Risk:04	Req:11	The Simulation is not Deterministic	Low	High
Risk:05	Req:13	The program can not meet it's runtime requirements, and takes too long to complete	High	Medium
Risk:06	All	I face technical difficulties, and loose access to my workstation	Low	High
Risk:07	Req:15	The Program is not intuitive to use	High	Medium
Risk:08	Req:16	The program experiences frequent crashes during runtime	Low	High
Risk:09	Req:09	Framework does not support updating of parameters during runtime	Medium	High

Chapter 2

Methodology

As is the structure for the master course at NTNU Gjøvik, much of the third semester is spent making plans for the upcoming master thesis.
So too was the case for Nationsim, which was originally started via a project plan, a tool study, and an early prototype in the latter half of 2021.
These all are the product of a full course, so they will be included as appendixes at the end of the thesis, should the reader be interested in them, and will occasionally be referred to as the reasoning behind various decisions made over the course of the project, particularly towards the beginning of the project in early 2022.

2.1 Technical Design

Originally, the project was meant to only run over 1 year, including the semester of courses that were used for a lot of the planning work.
As evidenced by the fact that we are now in early 2023 as I am writing this, that clearly did not work out.
The original project plan made the assumption that I would be able to effectively work for about 28 weeks and deliver an acceptable product that would ideally cover three separated modules, as well as a module combining their outputs into one output file.
With every module expected to be roughly the same complexity and size, each module was estimated to take about five and a half weeks of work to be able to be completed, with an additional five weeks for writing the thesis.
The long time spent on writing would have given me plenty of time to finish up any remaining work on the modules and debugging after the allotted time was over.

The work plan in it's entirety can be seen in Paper I, along with the rest of the project plan that was made alongside it for the course IMT4205 - Research Project Planning.

This plan however did not end up working out due to a variety of unforeseen circumstances arising in my personal life, as well as the original scoping for the project being more optimistic than anticipated.

The original plan saw the final product as a set of modules that would be only loosely coupled to one another and each simulates a separate part of the nations that make up the model.
Each of these systems would receive parameters from the other systems, and use these alongside its own parameters to simulate the different parts of a nation influencing the others, from the economy, to international relations, to internal politics.

Keeping these modules largely decoupled would increase the readability of the code base, as it meant not having to look at the totality of what would undoubtedly be hundreds if not thousands of lines of code, instead allowing each module to be viewed (and ideally debugged) separately.
Additionally, it also meant that modules could easily be turned on or off, as well as minimizing the number of touchpoints between systems.
Fewer touch-points would mean having to check fewer places for errors cascading through the system while bug-fixing, and it would mean that if a module could not be completed that less of the codebase would have to be altered in order to ensure the remaining modules still functioned at full capacity.

With each module remaining agnostic to the inner workings of the other modules, changing out modules at a later point would also be easier, as so long as the interface remained unchanged, no changes would have to be made in other modules to keep the program functioning.

As the original plan involved both prototyping and testing for each individual module, as well as strongly separated modules that were meant to function nearly independently, the requirements for a robust framework that could handle complex simulations at speed, which allowed me to maintain an overview of what no-doubt would become a complex architecture of interfacing modules was key for the implementation phase.
At the same time however, it became increasingly clear that my lack of experience in the field of simulation and social sciences would necessitate a lot of rapid prototyping to test out ideas and choose the ones that worked while being able to not too precious about the ones that did not.

Herein lies the problem, however, as ensuring that tools for testing, strong separation of files, and a focus on good performance would all be available, has a tendency to increase the complexity of the framework used.
The first major decision for the project thus became the decision to utilize two separate frameworks instead.
A lightweight, simple-to-use framework that would enable me to quickly test out

ideas in small simulations before incorporating them into the bigger project, and a larger, more complex but also more performant framework with capabilities for testing, which would be used for the actual implementation of those ideas.

As the field of Agent-based modeling frameworks is vast, however, I decided to do a preliminary study into the tools readily available to me in order to find the ones that made the most sense for my project and level of skill in the field.

2.2 Investigating Tools

Over the course of the course IMT4134 - Specialisation in Software Engineering, I investigated a variety of frameworks that seemed like they could be good fits for this project.
I knew from the beginning of the project that I wanted to go with an agent-based approach, as it seemed like the natural fit for the problem presented.
Being able to model each nation as its own, independent agent with its own impact on the world allowed the project to intuitively focus on the interactions between these independent nation agents.

Luckily there are many varying frameworks that have been created for intrepid researchers and hobbyists to use.
With the technical criteria of having to be publicly available free of charge due to a lack of funding for licenses being available for master projects, as well as being available on windows-based devices as that would be the devices I would be working on for the project, the list of frameworks was still only somewhat narrowed down.

Due to the findings presented in Paper II , I knew that Netlogo was very simple to work with, quick to get projects going, and surprisingly fast in its execution, despite how simple it appeared on the surface.
One of the veterans in the field, Netlogo had comprehensive documentation, plenty of online resources to get help from, as well as a well-developed community, and overall was the framework I felt was most easy to work with when it came to programming, despite the slightly arcane syntax that its programming language possesses.
However, Netlogo also suffers from a significant issue in that all code is normally written as functions in a single file, making the separation of various modules a purely conceptual one, rather than having different modules in different loosely coupled files.

On the other hand, Repast, the other standout framework in the study was a lot more cumbersome to work with, and occasionally turned out to be actually slower than Netlogo, but it was still more than fast enough to fit with the specific-

ation, and due to it being a java based framework it not only had access to testing through tools like JUnit but also provided a more separated, object-oriented layout for the project.

Caught between two frameworks that each would be ideal for specific parts of the project but less so for others, I took the decision of utilizing both at varying points during the project.
Since Netlogo allowed for the quick and intuitive creation of small models but got less and less understandable the more code a simulation had, it was ideal for the creation of rapid prototypes, which could help inform choices for the main implementation.

Repast meanwhile was much too cumbersome to set up for rapid prototyping in my experience but was nicely structured in a way that felt familiar to somebody that was taught primarily object-oriented programming, such as myself.
This meant that it was ideal for the full implementation of the project once the initial prototyping was done.
Each module could be created one after the other in prototypes in Netlogo, tested, and the best ideas sorted out, and then the entire module could be re-implemented into the full product in Repast.

2.3 Data sourcing

As the initial testing of how realistic the framework's output is relies on knowing some data from the real world, a source of data from real-world nations was needed, particularly in the fields of birth- and death rates, as these are a core part of the population growth function of the module handling the nation internal variables.
Any data that was not gathered via exploration of results, such as abstractions of more complex parameters like the cost of improving the infrastructure and technology used in producing a specific resource, or the starting values for the population and wealth of a nation, were instead gathered from the world Databank (https://www.worldbank.org/en/home).

It should be noted that the aforementioned website was used only to gather sensible boundaries for variables such as crude birth and death rates, boundaries which can be changed at will in either the interface of Netlogo or the script file for the scenario.

2.4 Re-evaluations

2.4.1 Framework choices

The upsides of using Netlogo and Repast for the project, one for prototyping and the other for the actual implementation are plentiful in their own right.
Easy rapid prototyping, closely related syntax, and the best of both worlds between a quick, lightweight coding framework for testing out ideas and a heavy-duty, neatly separated codebase that is easy to maintain and write tests for.
Overall, I would say that this is a major benefit to the quality of the code produced, provided the coder is familiar with both frameworks.
The end result, ideally, is a well-matured prototype that has gone through potentially dozens of iterations in a very short time, and a clean, well-tested, and maintained final implementation, completely separate from those prototypes.

The only downside of this would be that rather than being able to create the prototype, get the good ideas, and then simply implement that prototype directly, everything would have to be rewritten and the architecture redone for a separate framework that worked with a different layout.
Depending on who is asked, that might actually be a benefit as it ensures that the code is rewritten cleanly and properly documented, rather than having to clean up messy prototype code.
However, the process is a significant time-sink, and in the end, proved too time-consuming to be maintainable throughout the master project.

In the end, I had to make the choice of implementing everything in Netlogo, as the prototypes were well functioning by this point and only needed to be knitted together properly to function.
This did result in the codebase being entirely confined to one singular file, and the separation between modules was lessened somewhat, but it did mean that the project was able to get better results in the time that was allotted to it.

Theoretically, it would have been possible to write a Netlogo plugin to take care of some of the internal workings of nations, which could then have been stored separately from the main file, which would only have concerned itself with the market simulations and the interactions between the nations, however, Netlogo plugins are usually more used as means to the extent the capabilities of Netlogo with some new generic functionality that it is currently missing, like extended clustering algorithms, or like the CSV extension that is used for NationSim.
Additionally, this would have required even more additional time to learn how to create a Netlogo plugin, which would have gone against the whole purpose of switching to only using Netlogo.

Chapter 3

Implementation

3.1 Netlogo

With Repast no longer being a part of the project, all work focused instead on creating prototypes for various systems before merging them into the final project file.
Below is an overview of how the individual parts of the system function and interact.

3.1.1 Initialization

The first step for each model being run is the setting up of the initial variables.
This is divided into two distinct paths, depending on if the user provides a script for the initialization of the simulation, or if the simulation is meant to use randomized values based on parameters set in the interface.
The former of these two options require the creation of a script file, the setup of which will be discussed further in the section on scripting further on in this chapter.

The latter meanwhile utilizes the Netlogo interface Figure 3.1(left half) Figure 3.2(right half) for the project in order to provide the user some control over the simulation, though to a lesser extent than they would have with the script.

As we can see, the left of the interface is largely occupied by various variables, as well as the buttons to control the simulation itself.
The notable variables here are the runtime, which determines how many timesteps each simulation runs, the "times_to_run_the_simulation" input field, which determines the number of times that the simulation is re-run, the "source_file" user input, which allows the user to specify a file that contains a startup script for the simulation, the "debug" toggler, which when set to "on" will print out debug information during the simulations run, and the various variable sliders that allow

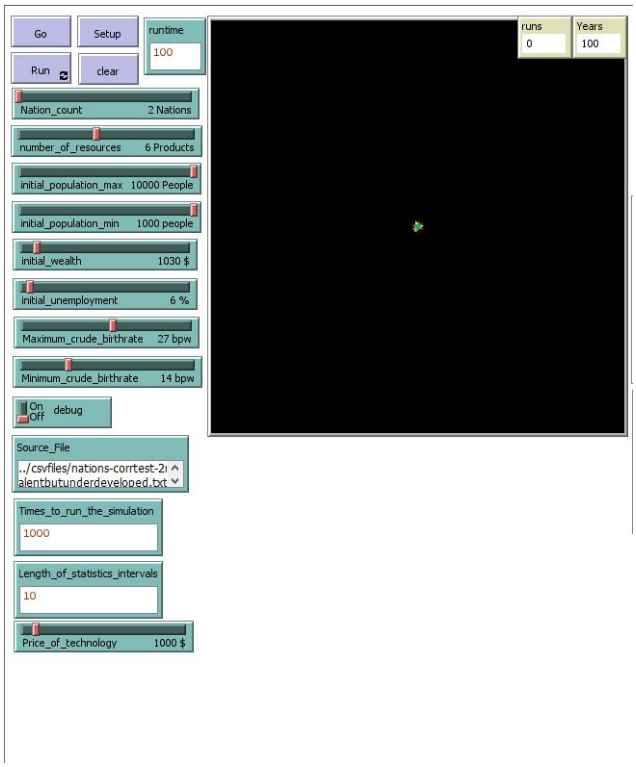


Figure 3.1: The left half of the Netlogo interface

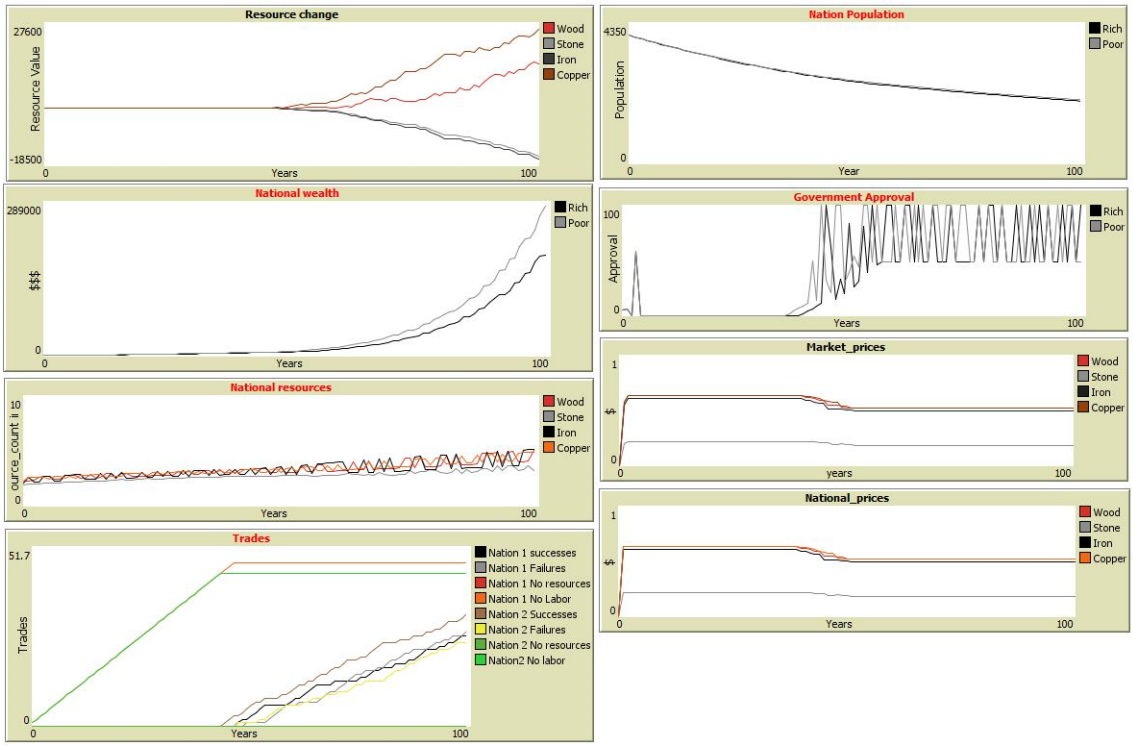


Figure 3.2: The right half of the Netlogo interface

the user to set the initial values for a variety of variables for each nation.

The simulation then first creates nations up to the number specified in either the file or the "number_of_nations" slider in the interface.

3.1.2 Government Module

Each nation additionally is initialized with a Government, which is an agent in its own right in order to comply with the low coupling ideals of the project.
The government holds various variables such as the level of taxation for the nation, the public funding that those taxes produce, and the budget division as to what that public funding is to be spent on.

The first thing at each timestep that the government module is responsible for is handling population growth alongside its own variables, as several key factors for population growth are stored and updated by the government module.

The population growth function (Code listing 3.1), in its most basic form, is an Euler-Lotka equation,

$$1 = \sum_{a=1}^{\omega} \lambda^{-a} \ell(a) b(a)$$

, which expresses the sum of the number of offspring born to a member of the population across its entire life.
In essence, it describes the number of people born via the number of women currently alive $\ell(a)$, multiplied by the number of births per woman $b(a)$. λ^{-a} is the discrete growth rate we are looking for.

The number of births per woman is the first thing that is calculated, by finding the impact that urbanization and education have on the growth of the population. GDP is also involved, but that is already factored into the death rate, so we exclude it here to make sure it does not doubly impact the population growth.
Finding urbanization presents itself as a challenge initially, as we assume a homogeneous population for the sake of the simulation, and have no real way to determine how the population is settled in our nation.
Instead, we need to look at the industries of our nation, and how large they are in comparison to one another.

Fishing, agriculture, mining endeavors, and other 1-st sector industries are typically not feasible in the constraints of a city and therefore lend themselves to more rural environments.
On the opposite side, second, third, and fourth sector industries typically rely on larger quantities of people being closely available, either as workforce or customer base, and therefore typically thrive in more urban environments.

By incorporating which sector each resource belongs to into our simulations resources, we can determine the level of urbanization of a nation by checking how much is being produced of first-sector resources and comparing it to the rest of production.
The larger a percentage the first-sector goods represent for our nation's total production, the more rural we assume the nation to be.

Education meanwhile is rather easy to find, as it is simply a matter of finding how much money our education budget represents compared to the size of the population.
With that, we can use an interpolation function to find how much impact our birthrate education has.

The number of women alive is calculated by first figuring out the number of deaths in the nation, which is done by calculating the relevant factors, such as the GDP, and the age of the population, which is calculated based on the assumption that a lower birthrate means that less young people are alive, and thus the average age of the population is higher.

With the birthrate already previously determined, we can figure out the impact of our GDP by checking how much money we are making in relation to our population.
With our factors gathered, we just need to multiply our population by our crude death rate and subtract that number from our population to get the remaining population.

Dividing this number by two gives us the number of our women, as we assume that on average one in two people will be female.
The number of females is then multiplied by our birthrate to get our total number of births.
Now, we just need to add our births and subtract our deaths from our original population to find our new population.

Code listing 3.1: Finding population growth

```
urbanization = sector 1 resource production vs sector 2 resource production
education = how much education we have per population
birthrate = urbanization + education impact

gdp = last income / population
deathrate = deathrate + gdp impact

deaths = population * deathrate

births = (population - deaths / 2) * birthrate

population = population + births - deaths
```

It also contains an approval score, which is determined by how well the nation

is being run according to the population.
This takes into consideration various factors, such as taxation level, economic growth, and education and welfare compared to the desires of the nation (Code listing 3.2).
These are found via interpolations, just as most of the factors in the population growth function.

Code listing 3.2: Updating approval

```
welfare impact = welfare budget vs ethics desire for welfare
education impact = education budget vs ethics desire for education

taxes impact = taxes * 2
negative impact = taxes impact + welfare impact + education impact

gdp impact = gdp growth this year

approval = gdp impact - negative impact
```

These factors are compared against each other to create a value between 0 and 100 which signifies how much the nation's population approves of the current government.
This value is currently not further utilized, as internal conflict, military presence, and the likes were scheduled to be added with the international relations module, which was meant to introduce a national military and all the economical and societal concerns that brings with it.

3.1.3 Economy Module

The economy module meanwhile is more firmly located with the nation agent, which contains all variables required for it.
The model for the economy that was chosen for the framework started off as a simple Ricardian model [19]. In a Ricardian model, two nations produce two resources and attempt to have the highest possible amount of both in accordance with their desires.
For this purpose, labor l is assumed to be homogenous, and able to freely move in between different industries.
A nation has a certain level of capability to produce one resource t, or more precisely, a value for how much of a given resource one labor can produce.
In Autarky, a nation is expected to produce a certain number of both resources in accordance with it's desires, as in the nation decides how much of its labor l should be used on each resource, and that produces a number of that resource as defined by the level of capability t described.
Then, we add an open market on which both nations trade their resources, where the worth of the resources may be different from the national wealth of the resources.

Now, every resource in addition to having a local value also has an international value.

By trading their resources, in essence by using their labor to produce resources for the open market rather than for domestic use, it should be possible for both nations to specialize in one of the resources which it:

- has a higher value of labor on the international market than domestically
- Is either better at producing than the other nation (t is higher), or least bad at producing (difference in t is smallest)

By both nations focusing on producing and selling this resource on the open market, both of them will be able to attain more of both resources compared to them working in autarky.

The Ricardian model thus can be seen as very optimistic when it comes to trading.
The model used for the simulation has a Ricardian model as its baseline, but expands on it quite a bit.
While the Ricardian model gives the value of each resource in terms of an opportunity cost for the other resource, the model used in NationSim utilizes concrete, monetary values for each resource, in order to facilitate more than two resources being traded more easily.

Every nation has a set of desires d for each resource that is defined at the start of the simulation.
This determines the initial domestic value of the resource V_n.
The international value of a resource V_i is then determined by taking the domestic values for all nations, and finding the median value of that list, ensuring that half the nations have a V_n higher, and half have a V_n that is lower than the V_i for that resource.

The value of labor is then calculated for each nation as $V_n * t$ domestically and $V_i * t$ internationally.
If $V_n * t < V_i * t$ then the resource would be a good choice for an export good, as the nation can make more money selling it than it is worth domestically.
Unlike the Ricardian model, however, this model does not make the assumption that trading is always beneficial.
Instead, it calculates the least amount of each resource acceptable to the nation, by multiplying the current population with the desire d for each resource.
It then calculates how much labor l_{req} would be needed to create that much of all resources via the formula:

$$l_{req} = l - \sum p * d_r / t$$

, where n is the specific resource.

If the remaining labor $R = l - l_{req}$ is below 0, so $l_{req} > l$ we need to recalculate our minimum desires, because we cannot fulfill them given the current state of the nation.
This is done by simply figuring out the percentage difference between the available and remaining labor l/l_{req} and multiplying every desire with this number before rerunning the previous calculation again.

If $R > 0$, that means we have labor left over after making the amount of the resource we need at the very least.
This labor can then be used to create profits.
But the simulation tries to ensure that the largest possible amount of profits is made and that it does by deciding whether to trade resources or produce them in autarky.

Our production in autarky is fairly easy to calculate, as it is always:

$$A = \sum (dp_n * R * v_n)$$

, where dp_n is the percentage of the sum of desires that d_n represents for resource n. So in cleartext, the formula means that the value created in autarky A is the percentage of labor used to produce resource n, which is determined by how much we want resource n, multiplied by the value of that labor.

In trade, things get slightly more complicated.
The first step is to find if there is a resource that is even worth trading away, which can be determined by if $V_n < V_i$ for that resource.
If no resource fulfills that criterion, the nation cannot make a profit by trading.
Otherwise, we choose the **most efficient** resource out of the list of possible export resources, as in the resource with the highest value difference $V_i - V_n$, and use all remaining labor to produce that resource.

This gives us our budget b, which is the amount of money we can use to buy resources from the international market.
The value gained this way T can be expressed via the formula:

$$T = \sum (b * dp_n)/V_{in} * V_{nn}$$

So the amount we buy of a resource is equal to how much of the budget b is used for each resource, as determined by how much it is desired compared to all other resources dp_n, divided by how much the resource n costs on the international market V_{in}.
Once we know how much we can buy of the resource, we multiply that by the domestic price of the resource V_{nn}, which gives us the value produced by that resource.
Summing these gives us the total value of trading our resources.

Now, in order to decide if we want to trade or just work in Autarky, we can determine if $A > T$. However, in order to cut down the computations required, I have created a formula that can check if the trade would be profitable without having to do the entirety of this calculation every time, by shortening down the formulas that produce A and T into one algebraic inequality that can be quickly solved to check if trading would be useful.

Rather than $\sum(b * dp_n)/V_{in} * V_{nn} > \sum(dp_n * l * v_n)$, we can compact the formula quite significantly.
First, we extract the most profitable resource to sell, our export good h. This gives us the formula: $b = V_{ih} * R$ and thus $T = V_{ih} * R/V_{ih} * V_{nh} + \sum V_{ih} * R/V_{in} * V_{nn}$.
Likewise for autarky we get $A = dp_h * R * V_{nh} + \sum dp_n * R * V_{nn}$.

The formulas for both Autarky and Trade can thus be shortened by removing variables that appear on both sides of the inequality sign. The final, shortened formula we arrive at is:

$$\sum V_{ih}/V_{in} > N$$

where N is the number of resources included in the simulation.
In essence, if we gather the differences between the international price of the export good and all other goods and if that number is higher than the total number of resources, the trade will be profitable.
AKA, if on average, the price of the export good is higher than the price of importing goods, the trade will create a profit.

By running this shortened formula as a test, we only need to find the most profitable export good and the international values of all goods.
As the most profitable export good is determined by our labor values, we can figure out whether a trade will be a success or a failure very early on in the calculations, saving us a lot of computations.
The shortened function has the pseudocode shown in Code listing 3.3.

Code listing 3.3: Nation trading Pseudocode

```
for each resource [
    if( Local value < market value)
    [
        export resources += resource
    ]
    minimum desired output = desire for resource * population
    required labor = required labor + (minimum desired output / output per labor)
]


if( required labor > labor)
[
    new desires = desires * (labor / required_labor)
    for each resource [
```

```
        new minimum desired output = new desire for resource * population
        new required labor = new required labor +
            (new minimum desired output / output per labor for resource)
    ]
]

if(labor - required labor = 0)
[
    No trade
    yearly production = minimum desired outputs
]

for each resource [
    Autarky production = minimum desired output + labor * (desire / sum of desires)
        * local value
]
if(length of export resources = 0)
{
    No trade
    yearly production = Autarky production
}

if( trade feasability test = true)
[
    budget = sell most valuable resource
    for each resource [
        resource budget = budget * (desire / sum of desires)
        Trade production = buy resource for resource budget
    ]

    yearly production = trade production
]
else
[
    yearly production = autarky production
]
```

3.1.4 Data Gathering

In order to ensure that all data that is relevant can be gathered, most data is stored as a list of values, which is appended by a new value at every time step.
This means in essence that each list has a length equal to the number of time steps that a run is meant to go.
For the sake of space, the actual output of the program does not print every value for these lists, but instead only prints the values at time-step 0, 10, 50, and 100.
In theory including every single value in the output is as easy as expanding the data collection functions a little, however, that would also proportionally increase the size of the output file.
I have chosen the values specified in order to have access to the starting value, a value shortly afterward, a halfway point, and the final value.
For now, the timestamps that are kept are hard-coded to be those numbers, but it would be fairly trivial to extend the framework to allow the user to choose which timestamps they want to be gathered for each resource.

3.2 Scripting

One of the major components of the framework is allowing users to upload their own data into the model to run simulations and scenarios they are interested in studying closer.
For this purpose, an alternative way to give the simulation input was added that allowed the user to write a script that would allow the user much more fine control over how the simulation and its parameters where initialized, as well as adding in custom scripts to be executed at certain points throughout the run of the simulation.

For ease of accessibility, a simple .txt file format was chosen, as that maximizes the number of ways that users can choose to write their scripts.
The example scripts come with some documentation that explains the general expected layout of data to be fed to the simulation, and the scripts accept a variety of different ways to control the inputs to the simulation, provided the datatype provided to the simulation stays the same, I.E. if a variable is expecting an input of type integer, it is possible to send a hard-coded integer value to the simulation, or a script evaluating to an integer, provided the correct syntax is followed.
The script file can by default accept integers, strings surrounded by quotation marks" and lists surrounded by square brackets [ ], as well as any combination of these, such as a list of lists.
As lists in Netlogo do not care about the types that make them up, a list can contain any combination of integers, strings, or lists.
Characters would simply be a string of length 1.

The script reserves a few specific characters for its syntax, that is characters that usually would not be found as a string of length 1 in a normal variable.
The scripts themselves explain these symbols in their documentation, but Table 3.1 also has an explanation for them.

With these operators, as well as the default input types, it is possible to script sophisticated behavior for the variables, such as updating certain parameters with a set interval between two thresholds, before moving on to the next parameter.

Additionally, the script supports the creation of "commands". These are read as a single number, which equates to what time-step of the simulation the command is executed.
On the next line, we include a "∼" operator, followed by a string that holds a command that will be run as though it had been typed into the command line of Netlogo at that specific time step.
All commands are then stored in a list and executed at the given timestep.
This allows for the scripting of custom events that will trigger at a predetermined

Table 3.1: Script Operators

Operator	**function**
"#"	Every value until the next occurrence of this operator is treated as a comment and ignored. Text still needs to be put into quotation marks in order to not cause a crash even if it is a comment.
"∼"	Reads the remainder of that line as a string and feeds it to the program.
">"	Everything after this operator is read into the program as a string and then evaluated as a reporter, with the result being the final value sent to the variable.
"|"	Indicates a predicate, and is followed by a string containing a true-false statement, which is followed by a list containing a value to be passed to the variable in case the statement is true, followed by a value to pass if the statement is false.
"z"	Indicates a predicate, and is followed by a string containing a true-false statement, which is followed by a list of lists that contains a number of values, our varying input. This is followed by another list of lists that function as a default value in case the statement evaluates to false. Next after that is a string that evaluates to the name of a variable, our mutator, and finally, another list containing a range of values for that variable, our thresholds . This specific operator is used for creating lists that have values that are changed as a variable hits certain thresh-holds, as noted in the last list. So for example, if the mutator chosen is "runs", then item 0 in the varying input list would be multiplied by how much bigger than item 0 of the thresholds our mutator is. Once the mutator reaches a value that is larger than item 1 in our thresh-holds, we move on to item 1 of our varying input, which is multiplied by our mutator minus the item 1 of our thresh-holds.

time in the simulation and is the cornerstone for more complicated simulations.

3.3 Data collection

As previously mentioned, all data is stored as a .CSV file at the end of the simulation.
For this purpose, a file is created where the first row is the names of all applicable variables, which varies in length depending on the number of resources and the number of nations chosen.
Every row below this first one corresponds to a single run of the simulation, with every column being associated with a specific variable that is collected during runtime.
These include values such as the wealth of a nation at certain time steps, the amount of a certain resource that has been bought or sold at that time step, or the random seed chosen for this set of runs.

The number of rows thus is determined by the number of runs that the simulation runs for, while the number of columns is equal to the number of variables. All data is separated with a ",", and can be loaded into a program such as Excel to be looked over directly, or to a data processing program such as R. The file itself is automatically generated at the end of the simulation and saved with a name that contains both the current date as well as the current time.

3.4 Deployment

Due to the nature of the project being a Netlogo file, the code can be run from the Netlogo framework by simply loading the project file, and operates directly from there. No further installation is required beyond that point.
As such, the only steps required in order for a user to get to run the framework is to first download the Netlogo framework, and then pull the current version of the Netlogo file for the project from https://github.com/Anarchydus/NationSim.

Then, all that remains is to open the .nlogo file for the project and either supply a .txt script to the program via the interface or just set the boundaries for the random initialization parameters and start the simulation.

Chapter 4

Research

This chapter focuses on the gathering of concrete data from the simulation in the Section "Data analysis", followed by explaining the data that has been gathered from our chosen scenarios to answer our Research questions.

4.1 Data analysis

To gain insights from the simulation, it was run through several data analysis tools, with the purpose of taking the somewhat archaic data lists, and turning them into something more humanly readable, in compliance with Req:05.
For this purpose, I used both the external Tool R in order to create several Correllograms, as well as the build-in visualization options of Netlogo in order to create several plots tracking the rise and fall of the variables I deemed most relevant.

4.1.1 R

The tool chosen for analyzing the data produced by the simulation was R, simply due to the fact that it was recommended by the supervisor for the thesis, and closer examination revealed it to be a fairly straightforward process to comb through data in the format that the program already created.
A few small adjustments had to be made to the framework in order to facilitate multiple runs being all put onto the same file, but the number of required changes overall was fairly low.

R allows the user of the program to quickly load the data sets produced by the simulation with a simple "read.csv" command. With the entirety of the data loaded in, all that remains is to decide which analysis is desired for the simulation and its data.
Due to R's ability to quickly write scripts that can be executed sequentially to create the desired output this is kept fairly simplistic.

For the purposes of Nationsim, one of the major things I was interested in was the creation of a Correllogram, showing the correlations between the variables that were being stored for each run.
This necessitates the simulation to be run a lot of times, mutating certain input variables between runs to see how they affect the whole.

The scripting data input for Nationsim facilitates this nicely, so several files were created that load a model containing one or more nations, either with the same values for their variables or with certain scenarios in mind.
From there, it is just about loading the file with the recorded data and using a script I created ahead of time to turn the data into a correlogram.

Code listing 4.1: The R script

```
filename <- readline(prompt="Enter␣filename:␣")

filepath <- paste(filepath,filename,sep = "\\")

df <- read.csv(filepath)
c <- cor(df, method = "spearman")
c[is.na(c)] <- 0
library(corrplot)
cp <- corrplot(c,type = "upper", tl.cex = 0.8, tl.col = "black",
addCoef.col = "black", number.cex = 0.5)
```

As we can see, the actual script as seen in Code listing 4.1 is fairly simplistic and meant to be run directly from RStudio in this case, though a better R programmer can probably also run it directly from the command-line.
The script asks for a file path that leads to the data output from nationsim, and then loads that file into the df variable.
From there, the data is turned into a correlogram, with the spearman algorithm since our data is continuous rather than discrete, and stored in the c variable.
All values that end up being NA values are values for variables that do not change throughout the runs, and therefore it cannot be determined if the variables influence each other.
In this case, we set that correlation to 0 in order to be able to correctly plot our correlogram.
using the corrplot library, we then create a plot, which can be exported as a png file.
Examples of such plots can be found below.

4.1.2 Correllograms

A correlogram is a diagram that shows the correlation of several variables, as in, it tracks if a change happens to one variable, which other variables are affected, and how.

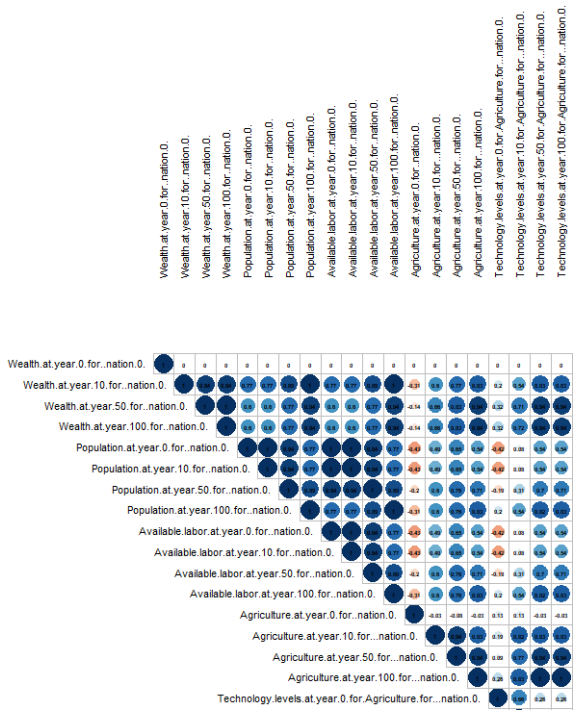


Figure 4.1: Small cutout of main correlogram

Ranks of correlation are stored as values between -1 and 1, with negative 1 meaning that as one variable grows in value, the other value decreases, and 1 meaning that if one variable grows, the other also grows.
By changing the variables of our simulation one after the other and storing the values for that run before moving on to the next, we can get a good idea of which variables influence other variables.
As a lot of the simulation is highly interconnected, we would expect to see a lot of correlations between variables.

However, as the average run with just 2 nations has about 350 variables being stored, these plots tend to be thousands of pixels in size in order to be able to see the names of the individual variables, as well as the ranks of correlation between them.
A full-size Correllogram is available in the repository for the project at https://github.com/Anarchydus/NationSim.

Instead, in this section I will present a small section of a correlogram for just a single nation in Figure 4.1, to give an overview of the scale involved in the diagrams.

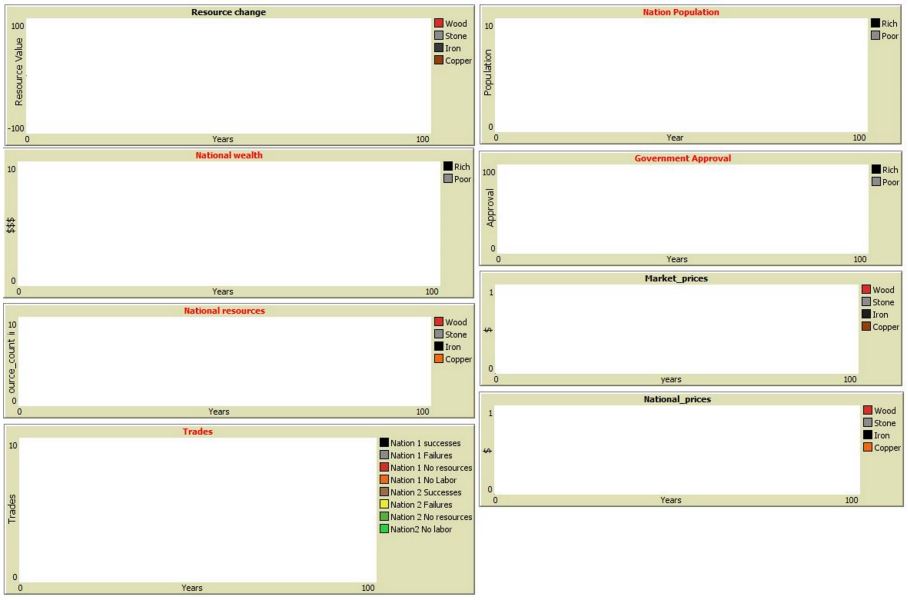


Figure 4.2: The plots of the Netlogo interface

4.1.3 Netlogo Plots

Another handy tool for the graphical analysis of data is the plots already built into NetLogo, which allow us to see the growth of various values in relation to each other.

For sake of readability, I have placed the plots on the right side of the interface, while most of the inputs are placed on the left.
The plots are easily creatable with only a minimal understanding of NetLogo and the codebase for Netlogo, so I should think it would be little to no problem to add new ones or alter the existing ones, even for a novice programmer.

The plots visible in figure Figure 4.2 represent, from top left to bottom right:

- How much of each resource has been bought or sold on the open market
- How the populations of each nation developed over the course of the run
- The national wealth of each nation
- How much the peoples of each nation approved of their governments
- How much of each resource was created by each resource at each timestep
- the international prices for each resource
- The number of trades performed by each resource
- the national prices of each resource

Additionally, we also have two monitors, one for the current time-step (Years) and one for the current run of the simulation, both of which are located in the upper right of the window which is usually used for visualization, near the center

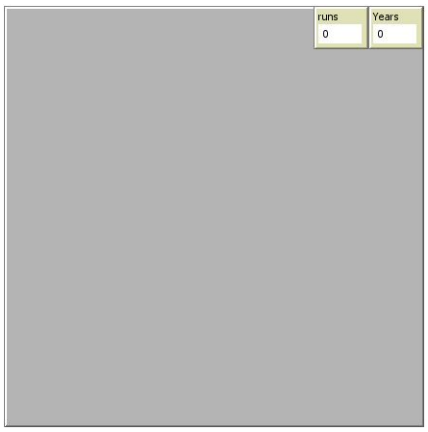


Figure 4.3: Netlogo center of interface

of the interface as seen in Figure 4.3.

These plots give a far less analytical but far quicker overview of how the nations develop over the course of each run.
On faster setups the simulation does however run the risk of finishing a run so quickly that the plots do not get a chance to render before being wiped again, however, so slowing down the ticks manually is highly recommended if looking at the plots' develop is desired.

4.2 Results

Our results are the concrete information that we have been able to attain from our simulation, that being the phenomena we can observe via our analyzed data.

4.2.1 Scenario 1

The purpose of Scenario 1 is that we simulate a nation that has a large impact on a particular resource on the open market, as well as two other nations that have a high desire for that resource but less ability to produce it, and then after the market stabilizes we remove the producer nation from the market and see what the resulting changes to the market and the economies of our consumer nations are.

Scenario 1 is made possible with the ability to script commands to be run at a certain time during the execution of a simulation.
Through that, we can signal the simulation to kill the nation in question at the halfway point of its runtime, and the plots created by Netlogo and R can give us

Figure 4.4: Prices affected by removed Producer

insights into what happens to the remaining nations and to the market.
[1]

The first interesting result of the nations disappearance can be observed in the plot which monitors the market prices of resources, Figure 4.4 , where we can see that at the halfway mark, the price of resource 1, here named "Wood" jumps significantly, with the price increasing from 0.3 to 0.5, a jump of over 66%
This is supported by the observation that in the plot for the average national price for the resource, the same significant jump can be noted, with the average national price being the determinant for the international price of a resource.
Several other resource values do however also increase by a non-trivial amount as a result of the disappearance of nation 3. These resources, resource 3 Iron and resource 4 copper grow with 33% and 11% respectively.

The graph for Trades performed by the two nations Figure 4.5 is also impacted, as we can see that the lines for successful vs unsuccessfully trades were on a trend to meet right around the line signifying the removal of the producer, only to be forced apart again as a result, recovering back onto their original trajectory by the end of the simulation.
These results paint a very clear picture of how the removal of a key player from the international market affects the other participants in that market.
The price of certain resources, particularly but not exclusively the one produced most prominently by the disappearing nation spikes, resulting in significant changes to the frequency with which a certain resource is bought compared to sold, and can even re-shuffle the economical power players in the market as nations that previously could not get their foot in the door suddenly can quickly gain more

[1] Unfortunately, the data gathering breaks a little if the number of nations specified in the script file does not equal the number of nations still present by the end of the simulation, resulting in columns being created for more variables than there are, and the variables of the market being written under columns labeled with the names of variables of the now non-existent third nation.
That is however not too much of an issue, as a lot of it can be fixed with simple image editing to return the correct names to the correct variables.
In an ideal world, this issue would be fixed, but with the time available the manual solution will have to suffice.

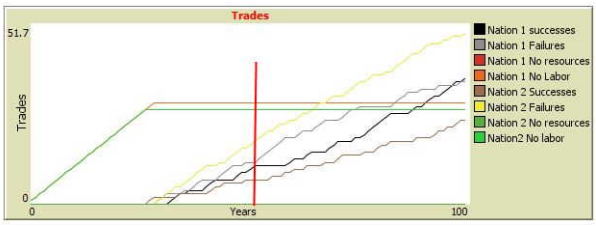


Figure 4.5: Graph of trades with point of removal marked

wealth.

Thus it can be concluded that major producers in the market being heavily impacted has cascading results to all other members of that market, with other producers of the same resource being able to break into the market, which might affect the production of other resources which need not be relied on any further by nations that might previously have exclusively produced that resource.

Not all nations win in this exchange, as the price of certain goods soars and others drops, the economy of other nations that relied on the state of the market begin to suffer.
However, over a longer time frame, we begin to see the market actually starting to stabilize, as the nations bounce back from the initial changes.
This two makes sense, as nations will eventually adapt to the new market landscape, and although the prices of the resources in question remain high, other nations have returned to making trades at the same frequency and have their wealth increased at steady levels again.

4.2.2 Scenario 2

The purpose of scenario 2 meanwhile is to give an insight as to how nations shape each-others economy even if one nation is significantly wealthier and more capable of producing resources than the other.
This is a fairly simple scenario to script since it just relies on making sure that the less advantaged nation always has lower values than the wealthier nation.
First, however, we require a baseline, so we first run the simulation with a few other starting scripts to get an overview as to how two nations interacting affect each other.

For this, we have a set of five scripts, one which holds two entirely identical nations, one which holds two nations that are equivalent but have different desires and abilities to produce, and one which holds two nations where one has significantly more starting capital.
The next set of scripts has two very similar tests, one which holds two equivalent

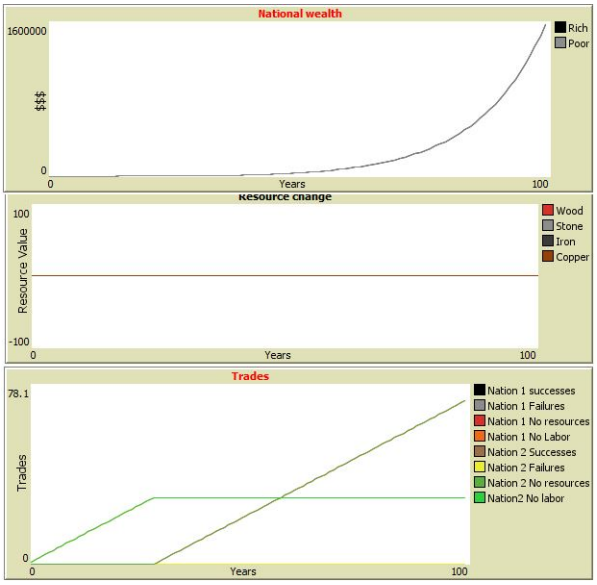


Figure 4.6: Wealth, Resources traded, and Trade record for Test 1

nations, one which holds two nations where one has a greater ability to produce resources than the other, and one that is much the same, except that instead of increasing the production ability of one nation, we reduce that of the other.

When I say equivalent I mean that the values in their lists of desires and technology levels are the same, but they are ordered differently.
So if nation 1 has a desire for wood equal to 0.5 and for stone equal to 0.1, nation 2 might have a desire for wood equal to 0.1 and for stone equal to 0.5. Same values, different order.

The tests produce five distinct yet notable results.

Test 1: Identical nations

Test 1, where we set up two nations that are perfectly equal with all starting parameters and desires set to be the same, results in two nations that unsurprisingly grow in the exact same way.
All graphs become perfectly equivalent lines between the two nations, with no difference between the two input nations, as seen in Figure 4.6.

There also is very little trade to take note of with both nations having essentially no reason to interact due to a lack of export goods that would be worth selling, seeing as how the international price is the average of the local prices.
With local prices identical, the international price is equal to the national prices,

and thus no trade is worth it.
The only significant change occurs as both nations begin to enter a state during which they have so little population left remaining that they can start producing resources in autarky, rather than being forced to just take the minimum output, as seen in the trade record in Figure 4.6.

During this test, the start-point for both nations' wealth is known, and the endpoint can be easily extracted from the wealth graph. For this specific run, with the starting parameters chosen in the script file, the nations grew steadily from a starting wealth of 1000 to a final wealth of 1 550 000.
This obviously is entirely ludicrous for a nation to be able to accomplish in 100 years, but that can be chalked up to resulting from the initial choice of parameters. The numbers still serve well enough to set a baseline.

Test 2: Equivalent Nations

The second test makes the nation's equivalent but not identical.
This test has quite a significantly different result to the one before as now trades are happening at a decent rate between the two nations, as they both have something to gain from selling and buying on the open market.
The results are initially very similar to the first test as both nations are as of yet unable to produce enough resources in order to fulfill their demands.
However, unlike in test 1, once the nations reach a state where they can begin to shoulder the burden of their own population's resource demands, the nations quickly begin trading their resources with each other.
Since they are not identical, one nation manages to perform slightly better than the other, giving us two separate graphs Figure 4.7 rather than the united wealth graph of Figure 4.6.

In this test, Nation 1 increased its starting wealth of 1000 to a total 1 840 000, showing a significant improvement over the number achieved in test 1.
Nation 2 however managed an even greater increase, reaching 2 350 000 in the same amount of time, likely due to being better at selling a resource that has a higher value compared to nation 1.

Test 3: Capital difference

Next, in the third test, we run the two nations from test 2 one more time, except now nation 1 has 10 000 starting wealth instead of 1000, while nation 2 has only 500.
Remembering back to test 2, we saw that nation 2 at the end accumulated 600 000 more wealth than nation 1 when both nations started off equal.
In test 4 however, we can see from Figure 4.8 that both nations actually end up

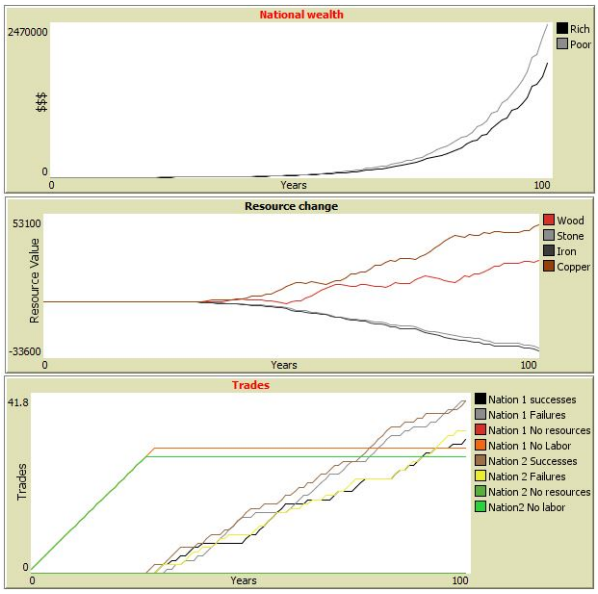


Figure 4.7: Wealth, Resources traded, and Trade record for Test 2

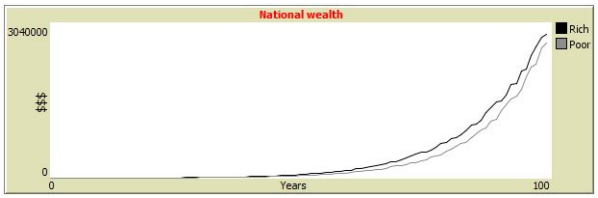


Figure 4.8: Wealth Graph for test 3

with near equivalent wealth.
Thus nation 1 has managed to mostly negate the massive disadvantage it had during test 2 with the additional starting capital, but nation 2 can still overcome its handy cap and reach nation 1's wealth via being better at producing certain goods.
This comes despite the fact that Figure 4.9 shows that nation 1 made more successful trades than nation 2 for quite a significant amount of time.

Test 4: One more developed nation

Again, we begin with one nation equivalent to its counterpart in test 2, this time with nation 2 remaining the same while nation 1 has all its production abilities doubled compared to the norm.
Unsurprisingly, this results in nation 1 having a massive advantage during the run. Once both nations start trading, nation 1 quickly grows, as seen in the wealth and trade graphs of Figure 4.10, until it finally reaches a value of 25 000 000.

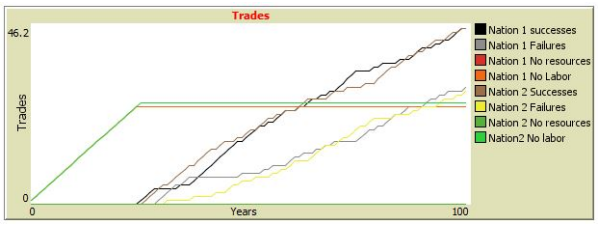


Figure 4.9: Trades for test 3

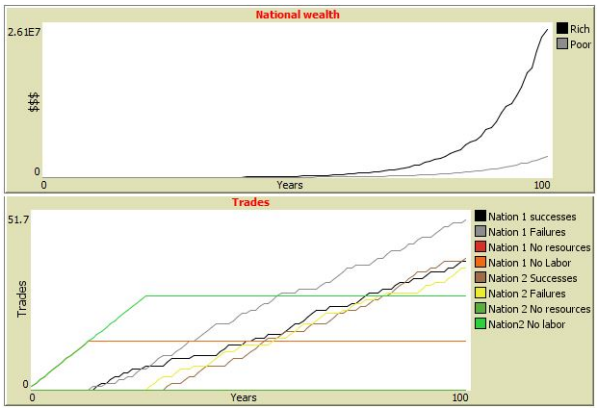


Figure 4.10: Wealth and Trades of Test 4

Nation 2 remains a lot tamer by comparison, but still manages to reach an impressive 3 600 000
Despite being completely identical to its equivalent in test 2, Nation 2 managed to turn almost double the profit over the same time span while trading with the vastly superior nation 1.

Test 5: One less developed nation

Finally, the fifth test showcases a scenario much like the fourth test.
This time it is nation 1 that remains at the baseline set in test 2, while nation 2 has its ability to produce resources cut in half, in order to simulate a more underdeveloped nation.
The results are also fairly interesting, as just like test 2 we see one nation ending up dominant, however, the difference is far more pronounced.
The nations still struggle towards the beginning to hold their weight against their own population's desires, before quickly beginning to avidly trade, however, as we see in the trade graph of Figure 4.11, Nation 1's ability to trade instantly outpaces and outperforms that of Nation 2, a trend that continues right until the end, when nation 2 begins closing the gap.

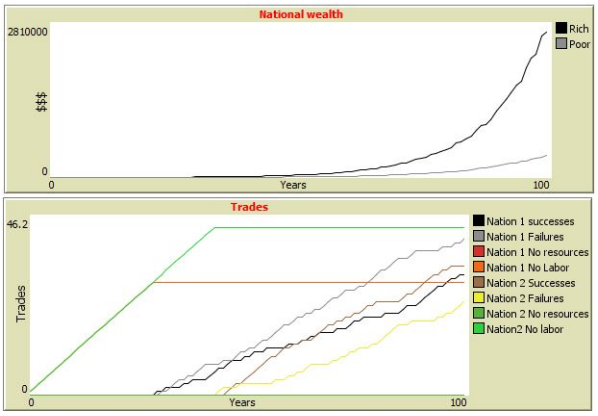


Figure 4.11: Wealth and Trade Graphs for test 5

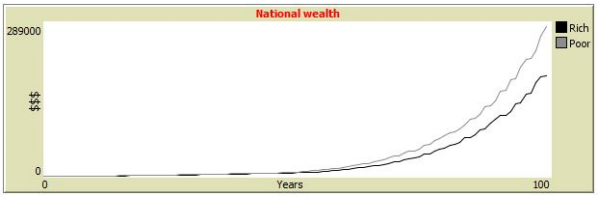


Figure 4.12: Wealth graph for two equivalent less developed nations

Likewise, the wealth graph in Figure 4.11 shows that Nation 1 clearly makes a lot more money than Nation 2 over the time tested, which should be expected. By the end of the simulation, Nation 1 has increased its 1000 starting cash to a total of 2 830 000, while Nation 2 has increased its own wealth only up to 400 000 from the same starting point.
While this may seem bad, comparing that number to a test where both nations are underdeveloped, Nation 2 still does impressively well compared to that test, where it only managed to earn 280 000, as can be seen in Figure 4.12.

Chapter 5

Validation and discussion

5.1 Validating Results

As there is no easy way of directly validating the data we have received against concrete, real-world data, nor can I do a full user test to determine whether the data produced is convincing and realistic, due to the limitations of privacy concerns, as well as the time and resources required to set up a proper user testing environment.
Therefore, we will rely on looking over the narrative that the data provides for us, the story that the correlogram and the plots tell, and then we compare that narrative to the real world to see if the narratives match real-world observations.

In essence, rather than testing the individual variables for their real-world validity, we aggregate them into a "big picture" where the *behaviors emerging* from the variables is what is validated against real-world data.
For this purpose, we simply look at the story that the data tells and then check if there is a real-world scenario that matches the simulated narrative.

5.1.1 Scenario 1

Scenario 1 shows the prices of several commodities soaring as the major producer is suddenly no longer able to participate in the market to the same capacity as they were before.
As a major producer is no longer able to participate in the market, prices greatly increase, and the economy of surrounding nations is impacted to varying degrees, with some experiencing a significant economic downturn.
The narrative here, thereby, seems to be "If a major producer for a resource is removed from the global market, several resources, in particular the one that was produced by the removed nation, experience severe increases in prices. This results in non-trivial changes in the economy of the nations that remain in the market, as trade patterns shift."

In order to verify this we need a real-world example of one or more nations being removed from the global market having a significant impact on the prices of several kinds of goods, particularly the ones that were most heavily produced by those nations, as well as significant economical disruption around the other members in that market.
In order to find such a scenario we have to look no further than the Russia-Ukraine war, which has heavily impacted the availability of resources normally produced in great quantities by the two nations, that being oil, gas, coal, and grain in particular. This has had a significant impact on the global economy, resulting in predicted economic growth being reduced by 1% globally(Orhan [20])

As the real-world narrative lines up very well with the narrative produced by the simulation, it would seem that the results produced are very much realistic to the real world.
If anything, the real-world data shows far more drastic increases in price than the simulation, which is most likely caused by the input data used to create the nations in the simulation does not line up with the real-world data for Russia, Ukraine, and the nations impacted by the changes in the international market.

As the output of the simulation closely represents the real-world scenario, I will conclude that for this scenario, the simulation can produce life-like results that are believably close to the real world.

5.1.2 Scenario 2

The testing of two nations and their performance, given certain starting parameters also yields some fascinating results.
The first test, pitting two identical nations against one another proves that the model correctly responds to two nations that have no reason to be trading with one another, outside of political reasons which are not yet modeled.
Barring any outside intervention, a nation has no reason to trade if the price of its goods on the international market is equal to the price of its goods on its internal market.

The second test showcases a clear increase in wealth when two equivalent nations are trading, compared to when those nations are not doing so.
This result is repeated across the board, suggesting that two nations trading end up making more money than two nations that do not do so.
In that regard, we find a return to the Ricardian model in a way, where trading is always the more beneficial option for a nation.
This is not too surprising, as that is the theory upon which the NationSim-frameworks economy simulation is based.
There is evidence to back up that participating in international trade stimulates economic growth, as can be seen in for example Poland.

With the fall of the Soviet Union in 1991, Poland was suddenly open to entering a free market economy and trading with the rest of Europe and the wider world. We can subsequently see a massive rise in GDP growth since that point, with increases ranging from 0.9 % to 7.1% between the years of 1992 and 2020 [21]

This could suggest that the model has good reason to claim that trading with other nations is beneficial for even the nation that makes less money in the equation.

Likewise, the results of tests 3, 4, and 5 all support that analysis, with nations making significantly more money than their equivalent nations in the test without trading, with the exception of test 5's nation 2, which of course has its production capabilities severely reduced when compared to nation 2 in test 1.
Tests 4 and 5 do still show however that in all cases when compared to the baseline of two equivalent nations trading, two nations with high inequality in their ability to produce goods stand to benefit more from trading with each other than their counterparts in test 2.

The control test for test 5 in which both nations have a reduced capacity for production still shows the same relationship between its own results and the result of test 5 that test 2 shows for test 4.
Even despite the fact that the nations are the same as those in the test where both nations were equivalent, nations trading with nations that are significantly more or less capable to produce resources stands to win more by trading with each other than by trading with nations of equivalent production capabilities.

However, the tests showcase a concerning trend where the more production-capable nation disproportionately benefits from the relationship compared to its less developed trade partner.
This matches with the data of [22], which showcases that since the early 1820s, while the GDP per capita in Europe has increased significantly, rising from around 2000 to 39 000 international $ between 1820 and 2018, a 1850% increase, Sub-saharan Africa only grew from 800 to 3 500 international $ over the same timeframe, an increase of just 337%.[1]

Finally, it is noteworthy that test 3 shows that a nation that is rich but is worse

[1]This somewhat clashes with the fact that GDP in western Europe has over the same timeframe risen from around 305 billion to 17 trillion, an increase of 5388% while Sub-saharan Africa's GDP has increased from 48 billion to 3.7 trillion, 7608%[23].
That relationship is not shown in the tests, but neither is the comparatively large difference in population growth over the same period of time, as population growth factors were not necessarily changed for the script.
Likewise, the model does not take into consideration foreign aid initiatives that might increase the ability of a poorer nation to develop and sustain higher populations than otherwise possible.

at producing profits through trade might be overtaken with time by a nation that is initially poorer but has better abilities for producing desirable resources.
Likewise, reducing the amount of money a nation has to invest into its infrastructure initially greatly impacts the amount of money that that nation makes over the course of the simulation.
This can potentially be linked to certain nations with a high ability to produce valuable resources gradually overtaking traditionally wealthy nations, like the rise of east-Asian countries such as China, which had a significantly lower GDP than Europe as recently as the 1970s, but has now surpassed the entirety of the European Union in terms of GDP [24].

5.1.3 Discussion

Despite being created as a framework that remains entirely agnostic to the purpose of the scenarios it is supplied with, NationSim manages to create complex interactions between Nations and their economic relationships that closely mimic those exhibited in real-life data.
By creating scripts that introduce specific circumstances to the nations interacting over the course of the simulation, a wide variety of different scenarios can be modeled, all without having to change the baseline framework.
Without any outside interference beyond the initially given parameters, the model can recreate complex relationships between nations and their economic co-dependencies, which create narratives that closely match the stories told to us by real-world data.

Since NationSim can recreate complex interactions between nations such as market crashes as a result of producers leaving the market as demonstrated in scenario 1, and the importance of trade for economic growth demonstrated in the development of nations such as Poland, as well as the importance of the ability to produce desirable resources vs capital, and the inequality in wealth growth experienced between highly developed and lesser developed trading partners, we can safely say that NationSim can model a wide variety of realistic interactions between nations.

Future tests

Since Nationsim was made to provide a generic model of nations interacting, there certainly are a lot more potential scenarios that could be tested by creating the appropriate scripts.
Besides the here provided scripts, NationSim has the capability of modeling nearly any scenario which impacts a nation's economy or population, as well as a variety of other factors.
Other ideas for potentially informative scenarios would be a Covid-19 scenario, where the population of nations is reduced via "Covid-wave" events at certain intervals, and seeing how this impacts the economies of these nations, as well as

other nations in the same market.
Also of potential use is the ability to alter the desires and production capabilities of nations, to simulate a nation transitioning towards a higher demand for a different good, or an ecological disaster that destroys the infrastructure of a nation.

5.2 Validating Requirements

NationSim sets out to accomplish a variety of tasks beyond the ones previously discussed, as "the simulation must run realistic interactions between a set of predefined nations" is only one of the 17 functional and non-functional requirements that were originally created to validate NationSim as a project, rather than just it's ability to model believable international economies and interactions between nations.

The previously explained capabilities of Nationsim already serve to confirm NationSim's ability to function as intended.
Requirement 1 is thereby fulfilled, and the above tests serve to confirm that Requirement 2 is also completed.

Requirements 3, 4, and 17 are fulfilled by the options to start the simulation either via the interface or the script, allowing users fine control over how the simulation functions.
' Likewise, the plots of Netlogo, as well as the CSV-files that are created and subsequently analyzed via the R script allow NationSim to also claim to have fulfilled Requirements 5, 7, 14, and an argument could be made that we fulfill Requirement 6, at least partially, as a user could technically read over the original input data as stored in the .csv file, and then feed that back into the simulation to recreate the same simulation again.

Requirement 8 is only partially fulfilled again, as while Netlogo allows the user to halt the simulation at any time by simply clicking the run button again and the user could then theoretically read the current data of the simulation and use that as input for a new simulation, that hardly seems as though that fulfills the spirit of the requirement.

Likewise, Requirement 9 is also only partially fulfilled, as the user can edit the parameters of the simulation during runtime, but only if that change is scripted as a command into the script file before the simulation is started.

Requirement 10 never ended up being implemented, as it was mostly present to simulate the physical distance between nations, which NationSim in its current state does not take into account any further.

Requirement 11 however is again present in the simulation, as all random variables are fed with a seed that is defined and recorded at the start of the first run of the simulation.
It is theoretically also possible to change the seed for every run via the script, allowing for simulations with randomized parameters to have different results for every run.

Netlogo makes the completion of Requirement 12 also fairly simplistic, as the present "tick"s can simply be defined to be one year in length, and time-dependent variables can then be defined in such a way where every time they are updated they do as if a year had passed.

After a cursory check with an online stopwatch and a quick random simulation, I have concluded that it takes the simulation roughly 2 seconds +- 1 second to run 1000 years for 10 nations, provided the user turns off the "view updates" option next to the speed slider.
The "timer" function, which measures the time since reset-timer was called right at the start of the simulation and after every run, is used to output how much time has passed once the run of the simulation finishes, outputs a time between 0.893 seconds and 0.96 seconds for one complete run of 10 nations for 1000 years.
As the original goal was 1 minute, I believe we can safely say that Requirement 13 is also taken into account.

Due to the lack of user testing, verifying Requirement 15 is very difficult.
I have however taken steps to ensure that both the scripting and the NetLogo interface explain a lot, and have given a lot of examples of functioning scripts in the GitHub https://github.com/Anarchydus/NationSim, to make it as easy as possible for users to understand what is going on.

The tests done for Requirement 13 also neatly verify that Requirement 16 is fulfilled, as in order to get 10 timestamps for how long it took the program to finish, I had to run the simulation 10 times for 1000 ticks with 10 nations, which means that all functions executed by each nation had to fire 10 times per tick, 1000 times per run, for 10 runs, giving a total of 100 000 computations in that test alone without any issues or crashes.
Theoretically, as there is no limit for how many years the simulation will run, it is possible that the growth function for either population or wealth eventually hits the limit for the largest number Netlogo can handle and crashes the program, but at that point, I believe we are well in the territory of "excessively taxing user input beyond the expected norm".

Chapter 6

Conclusion

6.1 Future work

6.1.1 International Relations

Due to time constraints, several originally planned modules for the project had to be cut, chief among which is the module handling international relations, diplomacy, and warfare.
The groundwork for this module was already laid in a few early prototypes that spoofed how such a module could function by keeping track of a relationship score for every pair of nations, similar to Axelrod's approach to cooperation between nations, but with a more complex set of criteria for how and when these relationships would change.

With international trade already modeled in the simulation, separating the nations into clusters of neighbors that trade with each other rather than one central market would have been nearly trivial, and trading could have been used as one avenue for improving relations, just as tariffs and tolls could have been implemented to model sanctions and similar real-world concepts, which might have harmed relations in the long run.

Via the already in place "ethics" variables which govern how much a nation values certain actions or political stances, modeling isolationist, globalist, and warmongering nations could have been implemented, with the original plan being to utilize data from The world Value survey (https://www.worldvaluessurvey.org/wvs.jsp) to create initial nation archetypes, as well as using the scripting files to allow users to utilize their own archetypes.

Finally, the groundwork for enabling warfare is already laid, all that is required is to separate the population into how much of a working force is present and how much of it is military, the cost of this on the economy, both in resources required to fuel the war machine and in profits lost due to a reduced workforce, and finally

the approval system could be used to model internal unrest that requires dealing with in some way, be that the government operating more closely in line with the will of the people, or additional military resources being spent on keeping the peace at home.

In reality, a lot of the frameworks already present systems are essentially prepared for the addition of the International Relations module, and the only thing required to incorporate it into the project is another month or so of work time.

6.1.2 Improved Scripting

The scripting file currently is highly limited in its capabilities by both what the program is willing to take as input for the various variables, as well as what the actual scripting "syntax" allows for.
This is due to the fact that a lot of the scripting syntax was created on a "what do I need right now" basis, resulting in the creation of specialized, often complex commands, such as the "z" command discussed in Table 3.1.

For a more flexible scripting experience, it would probably be best to either update the syntax of Nationsim's scripting with some more capabilities for less static inputs from the user or to directly switch to a proper scripting language, like Lua or Python.
Enabling the user to write slightly more complex commands than simple if/else statements, reading commands from string, and the list-mutator operations of "z", should make the number of possible scenarios that can be modeled with the framework less restrictive.

The "commands" system, which handles scheduled commands that interact with the simulation during runtime, does work fairly well already, as it essentially functions as a command line for NetLogo and can take any NetLogo code, at least in theory.
In practice, the limitations of the "run" commands are quite plentiful and a more sophisticated scripting language would probably be for the best.
A dedicated Lua extension for Netlogo is not currently available, but Python can be directly integrated via the python extension.

6.1.3 Improved Data collection

Currently, the data collection is quite static in what it picks up, not allowing users to specify which variables are of importance, and it only really takes into consideration simulations that run up to 100 time-steps, given its "0,10,50,100" data collection time-steps.
Allowing users to choose which data is important to gather is not as trivial, as

there are potentially hundreds of variables for every simulation.
Really, it would be better for users to create their own R script for surveying the data they specifically are interested in, rather than trying to create a checklist of which variables the user wants to look over in Netlogo.

Appending functionality to allow the user to choose what time steps are recorded when it comes to lists of how data changes over time should not be too difficult, as it is just about sending in more lists of time steps into the system via the scripts and slightly updating the initialization functions for the simulation.
This would also neatly solve the issue of the system only really looking at data up to time-step 100.

6.1.4 AI decision making

Another part of the project that had to be dropped due to time constraints, the original plan was to introduce a form of Q-learning to aid in national decision-making.
Since the Q-matrix could have been populated with the same data as the international relations module via national stereotypes data, this would have been a good fit for the project as a total, as the initial decision-making of a nation could have been decided via this nation-archetype data, and then as the Q-matrix updates its own weights, the nation's ideals and culture could have changed over time to reflect the changing of the times.

This could have given insights into which kinds of nations prosper compared to others, and what the mechanisms that cause a nation to change from one archetype to another are.
Additionally, it also would be a good way of ensuring that the simulation's decision-making is less static than the current one, which does not allow nations to make "mistakes", always taking the path that fulfills their goals.
Other forms of AI could also be a good fit, but the linking between the Q matrix and the National stereotype ideals seemed very intuitive to model.

6.2 Insights

6.2.1 Modeling Frameworks

Despite the fact that in the end, Nationsim ended up being finished as a Netlogo-only file, the process of developing via both Netlogo and Repast had such clear advantages that I would almost universally recommend it now that the project is over.
The ability to quickly prototype in Netlogo is such a massive advantage compared

to a more complex object-oriented framework that I cannot remember ever having had an easier time getting ideas for how systems should work within just a few days.
And the thing that holds NetLogo back, in my experience at least, is the lack of a strong, object-oriented structure for the agents being created and the systems that govern their behaviors.

Here is where Repast shines, simply due to the fact that it has good documentation, and that it uses almost the same syntax as NetLogo.
The Netlogo and Relogo programming languages are almost identical in syntax and functionality, with the exception that Relogo is an object-oriented language.
The separation of the project into compact files, the ability to associate variables with certain objects, rather than variable names being reserved across the entire project, and the fact that Netlogo and Repast have the same primitives in their agents, patches, and links makes it not just easy to convert code from Netlogo to Repast without having to massively rejig the architecture, but it also means that code remains readable even as the complexity of the project increases.

Not to mention the fact that JUnit and similar testing frameworks are easily integrated into Repast's java architecture, ensuring that testing the module and the integration of modules at each step of the project is a lot easier than in NetLogo as well.

With the one caveat that development time increases significantly with this method, according to the experience gained in this project, so long as a project puts more emphasis on rapid prototyping, clean architecture, and robust frameworks than on a strict schedule, the dual-framework approach functions extremely well.

A few flaws with both frameworks did however become noticeable as progress was made on the project.
The lack of several quality-of-life functions in Netlogo was somewhat annoying.
The lack of a proper clamping function necessitated a convoluted and hardly readable use of Min and Max functions every time a variable needed to remain between two values.

Additionally, while there is an equivalent to a switch statement, its documentation is hidden in the documentation for if-else, which is the last place a programmer that knows how to write an if-statement would visit in a documentation I imagine.
It also was only added in version 6.1 in 2019, which is recent enough that finding information for a switch statement did not result in any discoveries upon first attempting to look for it.
The first result on google as to how to do this is as recent as October 2021, which

was after I initially started looking into Netlogo.
Multi-case If-else statements are not something I am used to from any other language, but I suppose if it was a little clearer that this was a possibility it would be fine.

Also, the fact that NetLogo differentiates between "-1" and "- 1" is somewhat annoying, as is the lack of a dedicated for-loop, necessitating writing either forever loops with a break condition, or while loops with an internally updating variable.

Repast meanwhile seemed to have a few strange inconsistencies between its documentation and implementation, most jarringly that the non-directed links, which are supposed to create a directionless relationship between two agents, were not functional in the version of Repast that I received after downloading, with the implementation code that was clearly present in the documentation missing from the files in this version.
This necessitated me spending time creating directionless links myself, using the documentation as a specification to figure out how they are meant to work, and how links are implemented under the hood of the framework.

But yes, other than these strange inconsistencies, I really cannot recommend either framework enough, and a lot of these issues can be fixed with a bit of coding work on the programmer's side.
Still, these quality-of-life upgrades would be highly appreciated.

6.2.2 Contributions

Over the course of this thesis, I have highlighted my contributions to the field of both Agent-based modeling in general via the work process described in Chapter 2 , as well as to the field of social-simulation via the final product of NationSim.

The Process

The process of utilizing separate frameworks to on one hand tackle rapid prototyping with the help of an accessible ABM framework that offers tools to aid in quick implementation such as Netlogo, which has been used extensively throughout the project, and on the other hand to utilize a more heavy-weight framework with more utilities built-in, as well as native support for a more object-oriented workflow to aid in the separation of logical blocks throughout the created program, aiding readability and enforcing low coupling between modules.
A framework such as Repast symphony.

The prototyping section of an ABM project is aided by the fact that Netlogo and similarly light-weight ABM frameworks often do not require compiling between

test runs, making testing, tweaking, and outright scrapping and re-implementing parts of the project a very quick process.
Netlogo's uncomplicated syntax and lack of scoping capabilities also aid in this, as it ensures that less time is used focusing on semantics such as the correct ordering of objects and inheritance, and instead more focus is put on the conceptual problems that the program faces, which is what prototyping is for, after all.

Likewise, a sturdier framework such as Repast symphony is less capable of quickly testing out ideas, but the added functionality of easily integrate-able JUnit tests to ensure the validity of the output of different modules, as well as the object-oriented structure of Java allowing for strong separation of variables and functions between different objects, means that while the prototyping framework is ideal for quickly testing potential solutions to smaller parts of the project, the larger heavy lifter of the frameworks can aid in ensuring that all modules continue to function as expected once implemented, and makes keeping an overview of the code-base far easier.

Netlogo and Repast in particular are well suited for this sort of dual workflow, as they both use very similar syntax for defining and controlling their agents, which makes it easy to translate ideas from one into the other.
Repast being a java-based framework that can be worked with and extended upon in Eclipse and other Java editors is also a nice bonus, as it brings with it access to all the extensions, libraries, and tools that have been created for regular java programs.

The only downside that was found about the dual framework approach to ABM creation was the increase in time required, both to learn and transition between frameworks, to translate code from the prototype framework into an architecture that makes sense for the implementation framework, and to find workarounds for the few fields in which the two frameworks are not functionally similar.

I also have discussed several downsides of both Frameworks in general, such as the fact that Netlogo requires all code to be in a single file, and does not allow for scoping of variables, as well as the fact that parts of Repast Symphonies code appeared to not be completely implemented for concepts such as un-directed links between agents.
Likewise, I offered some frustrations at certain QOL features being missing from the frameworks, such as for-loops in NetLogo.

Overall, I would say that while using two frameworks introduces a heavy time cost to a project, as well as doubling up on the inevitable hassle of working around both frameworks' quirks, the benefits far outweigh the downsides.
For any project that either has an extensive timeline, values rapid prototyping, and concise implementations, or simply wishes to introduce some variety into the

process of coding an AB Model, I can highly recommend the Dual-framework approach.

6.3 The Simulation

Over the course of this thesis, I have both described the concrete function, as well as the applications of NationSim, a tool that aims to aid in the recreation and subsequent understanding of interactions between nations as agents.
While nowhere near feature complete, nor capable of robust predictions about future developments of nations or economies, the goal of the project this thesis is based around was the creation of a tool that could recreate the narratives we can see in real-world nations.

Whether that story be the crashing of a market as a result of a main producer being unable to participate in the market further, such as tested in scenario 1, the mutual growth of two nations with disparate industrial capabilities, via international trade between themselves, or any similar story that the real world shows us examples off in the market crisis in response to the Russia-Ukraine war, or the growth of Poland's economy after entering the global market during the post-USSR era.

These are all stories that the program NationSim is capable of adequately recreating via a script of initial data that is used to create the world the user wishes to look at the nations interacting in, from the initial wealth of nations creating economical giants and industrious underdogs to the intial populations, or even the values of nations.
All of these can then be altered, and entire nations can be wiped from the map via scripted commands that can be injected into the program at run-time and can completely change the course of the simulation, creating an all-new story.

NationSim is not necessarily capable of quickly and easily providing answers to *why* those phenomena occur, but it *is* very much capable of reproducing them via a complex simulation of nations interacting with each other and an international market, allowing users to create a set of "what if" scenario's and testing them via a framework that produces narratives of nations interacting that are realistic enough that they match real-world occurrences.

The truly dedicated can always look through the variables that determine the behaviors produced by NationSim and work to find the correlations between them and the final narrative produced by Nationsims many plots and vast collections of data points, which are made available via a .csv file that can be easily analyzed by f.ex. the provided R-script.
This data sheds light on the connections between abstractions made to real-world

trade relations, and the final output of the simulation.
But really, the project this thesis documents was about creating a tool that could create a realistic story from user-defined initial circumstances.
The fact that the stories NationSim produces closely resemble the narratives created by real-world data means that the project has succeeded in its goals.

The requirements that were set for NationSim at the beginning of the project have all been tackled to varying degrees of success, and the research questions posed at the beginning of the thesis have also been answered.

The thesis concludes that the consequences of a main producer being removed from the market are a sharp increase in the market price for the main good produced by the removed nation, as well as a smaller but still significant increase in price for several other goods, as well as a non-insignificant impact upon the number of successful trades performed by other nations that slot into becoming the main producer for the resource in question.
This closely matches the observed changes in the global market as a result of the Russia-Ukraine war, which reduces both nations capability to participate in the international market.

The thesis furthermore shows that NationSim produces results indicating a positive change to wealth between nations that do trade on the open market and nations that do not, which matches data of Poland's GDP growth after entering into the global market economy after the fall of the USSR.
Additionally, NationSim showcases an inequality between the economic growth in trade-partners showcasing severe differences in infrastructure, which matches the development of wealth in European nations compared to their trade partners in Africa.
Finally, NationSim proposes that Nations with lower initial capital but greater ability to produce highly valued resources will eventually close the gap created by their lack of ability to expand their infrastructure, which matches the growth of the Chinese industry when compared to traditional economic powerhouses like European nations.

With our research questions answered with these stories produced by NationSim, the final contribution of this Thesis to the field of Social sciences is the link to the repository from which NationSim, its pre-made scripts, and an applicable example of an R script can be downloaded so that the readers of this thesis too can use NationSim to answer their questions or generate their own narratives that might match with real-world stories.
The Repository is publicly available here at https://github.com/Anarchydus/NationSim.

6.3.1 Final thoughts

Nationsim as a project has consumed the better part of the last year of my life, and another good chunk of a half year before that.
The project has been a fascinating, eye-opening experience about the complexities of Social-sciences that I previously had no investment or interest in, as well as a clear reminder of how important it is to know what exactly you are getting yourself into with a project.

A half a year of extra work to finish the master project compared to what was originally planned should be a good reminder of that last part.
But it has left me with a renewed appreciation of the field of agent-based modeling, the intricacies that actually go into making a functioning economy, and just how vast the field of social simulation actually stretches across topics.
What started off as a tool to help writers in creating the stories they wanted to tell has matured into a program that now tells stories of its own.

As somebody that sees themself as a storyteller as much as a programmer, I could not be happier with that outcome.
Creating a program that is capable of telling stories of its own is more than I had hoped for when I first started this project.

Unfortunately, every good story must come to an end, and this is the end of my story that is this thesis.
With this work, I hope to inspire others to take up the mantle of a storyteller and take on the challenge of trying to create something that can help tell and understand the stories that can puzzle people in their day-to-day life.
Maybe that is by creating new scripts for NationSim to test new scenarios, and maybe that is to create a similar program all on their own.
In the end, we all can only tell the story we want to tell, it is up to the audience what they take from it.

As my final, final thought, I would like to thank you, the reader, for taking the time to read through this story of mine, and I hope you find many more stories to hear, and maybe even some to tell.

Bibliography

[1] N. Gilbert and K. Troitzsch, *Simulation for the social scientist*. McGraw-Hill Education (UK), 2005.

[2] J. W. Forrester, 'World dynamics,' *Cambridge Mass.*, 1971.

[3] P. Kolesar, W. Walker and J. Hausner, 'Determining the relation between fire engine travel times and travel distances in new york city,' *Operations Research*, vol. 23, no. 4, pp. 614–627, 1975.

[4] D. L. Meadows, W. W. Behrens, D. H. Meadows, R. F. Naill, J. Randers and E. Zahn, *Dynamics of growth in a finite world*. Wright-Allen Press Cambridge, MA, 1974.

[5] G. H. Orcutt, J. Merz and H. Quinke, *Microanalytic simulation models to support social and financial policy*. North Holland, 1986, vol. 7.

[6] T. Toffoli and N. Margolus, *Cellular automata machines: a new environment for modeling*. MIT press, 1987.

[7] J. Doran, 'Foreknowledge in artificial societies,' in *Simulating Social Phenomena*, Springer, 1997, pp. 457–469.

[8] H. A. Simon, 'The sciences of the artificial mit press,' *Cambridge, MA*, 1969.

[9] J. D. Farmer and D. Foley, 'The economy needs agent-based modelling,' *Nature*, vol. 460, no. 7256, pp. 685–686, 2009.

[10] L. Tesfatsion, 'Agent-based computational economics: Overview and brief history,' 2022.

[11] S.-H. Chen and C.-H. Yeh, 'Evolving traders and the business school with genetic programming: A new architecture of the agent-based artificial stock market,' *Journal of Economic Dynamics and Control*, vol. 25, no. 3-4, pp. 363–393, 2001.

[12] L. Tesfatsion, 'Structure, behavior, and market power in an evolutionary labor market with adaptive search,' *Journal of economic dynamics and control*, vol. 25, no. 3-4, pp. 419–457, 2001.

[13] L. Tesfatsion, *A New Swing-Contract Design for Wholesale Power Markets*. John Wiley & Sons, 2020.

[14] B. LeBaron and L. Tesfatsion, 'Modeling macroeconomies as open-ended dynamic systems of interacting agents,' *American Economic Review*, vol. 98, no. 2, pp. 246–50, 2008.

[15] D. World, 'Longevity: Extending life span expectancy.,' 2022, Retrieved January 12, 2023. [Online]. Available: www.disabled-world.com/fitness/longevity/.

[16] J. P. Mackenbach, J. R. Valverde, M. Bopp, H. Brønnum-Hansen, P. Deboosere, R. Kalediene, K. Kovács, M. Leinsalu, P. Martikainen, G. Menvielle *et al.*, 'Determinants of inequalities in life expectancy: An international comparative study of eight risk factors,' *The Lancet Public Health*, vol. 4, no. 10, e529–e537, 2019.

[17] A. Pourreza, A. Sadeghi, M. Amini-Rarani, R. Khodayari-Zarnaq and H. Jafari, 'Contributing factors to the total fertility rate declining trend in the middle east and north africa: A systemic review,' *Journal of Health, Population and Nutrition*, vol. 40, no. 1, pp. 1–7, 2021.

[18] N. Gilbert, *Emergence in social simulation in gilbert, n. and conte, r.(eds.) artificial societies*, 1995.

[19] A. V. Deardorff *et al.*, 'The ricardian model,' Research Seminar in Internat. Economics, the University of Michigan, School . . ., 2007.

[20] E. Orhan, 'The effects of the russia - ukraine war on global trade,' *Journal of International Trade, Logistics and Law*, vol. 8, no. 1, pp. 141–146, 2022, ISSN: 2149-9748. [Online]. Available: http://jital.org/index.php/jital/article/view/277.

[21] D. W. Bank, 'Gdp growth (annual %) - poland,' Accessed: 15.01.2023. [Online]. Available: https://data.worldbank.org/indicator/NY.GDP.MKTP.KD.ZG?locations=PL.

[22] O. W. I. Data, 'Gdp per capita, 1820 to 2018,' Accessed: 15.01.2023. [Online]. Available: https://ourworldindata.org/grapher/gdp-per-capita-maddison-2020.

[23] O. W. I. Data, 'Gdp, 1820 to 2018,' Accessed: 15.01.2023. [Online]. Available: https://ourworldindata.org/grapher/gdp-world-regions-stacked-area?country=Sub-Sahara+Africa~Latin+America~Middle+East~South+and+South-East+Asia~East+Asia~Western+Offshoots~Eastern+Europe~Western+Europe.

[24] D. W. Bank, 'Gdp (current us$) - european union, china,' Accessed: 15.01.2023. [Online]. Available: https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?end=2021&locations=EU-CN&start=1960.

Paper I

The Original project plan for the creation of NationSim, as completed in the IMT4205 - Research Project Planning course, here included to help illustrate the original assumptions made for creating the project and to help explain the decision making in parts of the thesis.

IMT4205 Research Project Planning Report

Leon Cinquemani

NTNU
Norwegian University of
Science and Technology

Use this class with options
“medieteknikk,” “mediatechnology,” “informasjonssikkerhet” or
“informationsecurity”.

Use this package with options
“medieteknikk,” “mediatechnology,” “informasjonssikkerhet” or
“informationsecurity”.

Abstract

As an aspiring writer, or as anybody who creates fictional settings, aka fictional worlds, as a pass-time or for their job, the creation of a world's story is a daunting task. With potentially thousands of years of history needing to be filled out to gain a clear understanding of the political landscape, the history of nations and their people, and the state of the world as it stands, the creative workload can quickly become impossible to manage.
In this paper, I propose a tool to aid writers in creating historical data for their fictional worlds, by providing an agent based simulation which will run on a model comprised of the nations of said fictional world, and simulate their interactions over the course of history to produce historically significant events.
This simulation aims to produce a list, or set of lists, of these events that can serve as inspiration for the author, and work as a scaffolding upon which the more in-depth parts of the setting can be erected.
Thus I hope to reduce the creative load required to flesh out a Setting with a full history, as well as provide inspiration to the creators of such settings.

Contents

1 Introduction

1.1 Background

The designing and development of a fictional setting is an important step in a variety of different disciplines. A setting can be defined as the overall scope of the world a story takes place in, which could range from a single city to an entire universe. The word setting in this paper is taken to mean a fictional world in which a story takes place.
From writing fictional texts to creating video-games, or running tabletop role-playing games as a hobby, the creation of an immersive fictional world is an important part of each of these and other professions.

With Adult-fiction books alone generating revenues of over 4 Billion USD in 2017[13] in the US, video games, in general, generating a further estimated 108 Billion [9] USD worldwide, and the most popular Virtual Tabletop RPG playing platform, Roll20 has reportedly 8 million users worldwide[4], which does not even include people that enjoy Tabletop RPGs offline.
As such it ought to be clear that the market for fictional works is generating billions of USD every year, and that they are enjoyed by millions of people worldwide. As such the number of fictional worlds that are present across entertainment is steadily increasing, from multi-million dollar classics with massive legacies, such as "middle-earth" from "The Lord of the rings", to more recent works such as "Westeros" from "A Song of Ice and Fire", "The Forgotten Realms" from "Dungeons and Dragons", and the "Galaxy far far away" from "StarWars".

With the legendarily in-depth story of "Middle-earth", the lifework of J.R.R. Tolkien, spanning nearly 38000 years of history[5], which contain hundreds, if not thousands of noted events across the entire lifespan of the setting, it becomes clear that such an undertaking requires herculean effort on the author's part, as thousands of years need to be filled with societal changes, historical events, wars, the rise and fall of nations, and the general marching on of time.
Of course, few authors can boast to have as in-depths a setting as J.R.R. Tolkiens life work. I would even reason that most settings do only really go back a few hundred years, at worst a couple of thousands of years when it comes to how much of the history of the world is explored and taken into account for the story being told.

Therefore, I will in this paper propose the creation of a novel tool to assist writers in completing this task, by implementing a simulation in an Agent-based modeling framework, which will allow intrepid writers to model their settings nations, and will then simulate decades, centuries, or even eons of historical interactions between these nations.
Agent-based modeling and simulation (ABMS) is a discipline of simulation, which focuses on modeling a system from the bottom up[7]. That is to say, in an ABMS approach,

a researcher creates several "Agents"; data structures that correspond to individual units that make up the model, which could represent people, cars, or other, more abstract concepts.
Each agent has a variety of sensors and actuators, which allow it to perceive the state of the environment it is in, and autonomously choose to act using its actuators to influence the said environment, based on a variety of predefined criteria.

1.2 Definitions

Agent-based Modeling has been a staple for use across a variety of disciplines for decades by this point, and it makes sense to use in this scenario, as the nations of a world neatly can be modeled as agents, and as interactions between nations, environment, and other factors can not always be easily expressed via a mathematical model, it provides an intuitive way to study the history of the setting.
Then the tool will produce outputs based on the results of the said simulation, which will allow the author to gain a quick overview over significant historical events, that is to say, noteworthy changes to the nations, such as famines, wars, and shortages, as well as factors that may have contributed to these events.
This information can then serve as a starting point for writing more detailed reports of said events, including historical figures and the precise circumstances leading up to the event, and unfolding during it.

An Agent-based Model can be built entirely from the ground up, but it is far more common to use one of several available frameworks which offer users a suite of tools to make the initial creation of the model easier.
These come in a variety of forms, as discussed by Abar et Al[6], each being implemented for different platforms, via the use of different programming languages, and to varying levels of specialization for solving one specific kind of task.
The Frameworks of choice for this project will be discussed later in the feasibility study section 3.1, a choice which is substantiated by a previous study done by the author of this paper as part of another course, which will be included in Appendix A, purely to provide more context to the choices made.

1.3 Research Questions

It is rather difficult to condense the problem at hand into one or more concise research questions, as the project itself is focused largely on the implementation of a novel tool to aid in a process. Unfortunately, I can not use a question such as
How much can the time/effort required to create a fictional setting be reduced with the help of Agent-based Simulation?
, as that would be incredibly difficult to measure, especially since I would need to survey authors, game developers, and TTRPG players to find good metrics for this, which is outside of the scope of the project due to time constraints and privacy concerns.

Likewise, questions such as:
Can a World Model simulation be used to create compelling narratives from which a story

can be extrapolated?
Are too hard to find good metrics for, since how compelling a narrative is, is entirely subjective, and whether or not such narratives can be used to extrapolate a story is also not quantifiable, and thus cannot be measured easily.

The best traditional Research question that I have found for the project to answer is thus as follows:
To what extend can a World Model Simulation be used to create a believable set of interactions between nations, which are used to create a timeline of historical events, from which events of interest can be extracted?
This question has the benefits of allowing us to measure a variety of things, these being for example:

- How believable are the interactions between the nations of the model?
- How well can the timelines produced be used to find events of historical importance?

Both of these questions can be measured using concrete metrics, such as the similarity between interactions measured in the model and interactions observed in the real world, and the time taken to determine the historical importance of events.

1.4 Related works

While numerous papers are using agent-based modeling as a tool for Political[10], economical[14], and social sciences[7], the combination of these fields into a full Global Model[8] from a variety of contexts is much rarer, though not unheard of. Several of the features intended to be included in the finished model are already well researched, such as the simulation of international conflicts, which has already been tackled in papers such as the study by David P. Masad[11], demonstrating the power of agent-based modeling as a tool for understanding the circumstances that drive nations into conflict, as well as the behaviors noted around these times. Likewise, the paper by Thomas B. Pepinsky[12], defines several key theories on international relations and discusses implementations of models that focus on simulating these theories in action.

This goes a long way to prove that the individual components of the proposed system all already exist, having been researched and implemented throughout a variety of projects over many years. This is promising, as it means that I will not have to walk entirely un-trodden ground in the creation of the proposed system, and can draw inspiration from previous works that have implemented similar systems.
However, to the best of my knowledge, while there certainly are global models that go over nations in as much, or deeper depth than what I am intending for the tool suggested, the idea of using Global modeling and history simulation as part of a writing tool is a novel concept yet to be explored.

2 Requirements and Risks

2.1 Users

For this project, I will be considering a variety of users from the three primary target groups presented in chapter 1, as well as some secondary user groups. These groups are:

- Primary:

  1. Authors
     Authors create large, intricate Settings for the stories they want to tell, as they usually go a lot more in-depth into the history of the world they are writing about than the other categories. As such, Authors require large amounts of historical data to draw from when creating their settings, but generally, they have more time than the other two main groups when it comes to their deadlines.

  2. Video Game developers
     Video games often have tighter deadlines for their writing that authors might have in their works, and as such have to often compromise on the scope of the historical data created for their settings to only what is specifically relevant to the game being created. This of course is not a general rule, and Game developers might still find themselves in the need of an extensive history for their settings, but for this project, the Video-game developer represents the middle-ground between the Author and the TTRPG player when it comes to the amount of data required vs the speed at which it must be made available.

  3. TTRPG players
     Compared to their counterparts TTRPG players creating their settings often have a far more agile workflow when it comes to the development of their setting, as the deadlines for having to present the history of their worlds are a lot tighter, sometimes down to single weeks or even days between deliveries, but the amount of history that needs to be presented is also far smaller for these situations.
     This means that a TTRPG player might require historical data for his setting very quickly, but only in small amounts, as to not overwhelm their fellow players.

- Secondary:

  1. Hobby-Historians
     A hobbyist-historian might be interested in using the tool proposed less for its primary purpose of creating a set of historical events, and more for its potential for taking in data from real-world history and manipulating the parameters of these to create "What if?" scenarios.

2.1.1 User Stories

Label	User type	Description
US:01	Author	As an author, this user wants to be able to run the tool to create a full history of a set of nations to use as scaffolding for his next novel setting.
US:02	Author	As an author, this user wants to be able to fine-tune the simulation to run an exact number of years equal to the age of his setting.
US:03	Author	As an author, this user wants to be able to select the names of the nations in the simulation to match those in his setting to make them easier to differentiate at a glance when looking over results.
US:04	Author	As an author, this user wants to be able to separate nations that reasonably should not be able to get into contact with one another during the simulation.
US:05	Author	As an author, this user wants to be able to store the results of the simulation for extended periods to be able to divide the time he works on them over a large timeframe.
US:06	Author	As an author, this user wants to be able to influence the simulation to change the behavior and starting parameters of each nation to represent the different races and cultures of his setting.
US:07	Videogame developer	As a game developer, this user wants to be able to run the simulation in a reasonably short time, to meet his deadlines.
US:08	Videogame developer	As a game developer, this user wants to be able to have some control over how the different nations develop during the simulation to facilitate the story of the game.
US:09	TTRPG player	As a Table-top RPG player, this user wants to be able to alter the setting later down the line to facilitate a new idea, or to incorporate some input from his fellow players.
US:10	TTRPG player	As a Table-top RPG player, this user wants to be able to break the results from the simulation down into small chunks to be developed further as needed.
US:11	Historian	As a hobbyist-historian, this user wants to be able to import historical data for real nations into the framework to create what-if scenarios.

It should be noted here that a lot of these user stories could work for several of the user groups. I decided however to only include them in the first category they came to mind in, as it would not be productive to repeat similar, or equal user stories.

2.2 Requirements

Requirements will be rated by how crucial they are for the system to be valuable to the above-discussed user groups overall, and will be prioritized during the system's creation accordingly.
The list of requirements is by no means exhaustive, but as initial prototyping is currently still ongoing, and as the project is expected to run in a somewhat agile format, I expect more to be discovered over the course of the project.
All requirements are linked to the User story that they hope to fulfill via that User stories label.

2.2.1 Functional Requirements

Label	Related User story	Name	description	importance
Req:01	US:01 & US:02	Runtime	The program must run a full simulation of a set number of years	High
Req:02	US:01 & US:03 & US:06 & US:11	Realism	The simulation must run realistic interactions between a set of predefined nations	High
Req:03	US:03 & US:04 & US:06 & US:11	Adaptable Parameters	The Nations starting parameters need to be able to be influenced by the user	Med-High
Req:04	US:09 & Us:11	File loading	The Simulations parameters need to be able to be populated from pre-existing data	Med
Req:05	US:01 & US:05 & US:10 & US:11	Output Readability	The program needs to output it's results in a format that are easily human readable	High
Req:06	US:09	Output reuse	The program needs to output it's results in a way that allows it to be fed back into the simulation as starting parameters	Med-Low
Req:07	US:01 & US:03 & US:05 & US:10 & US:11	Auditing	The user has to be able to draw conclusions from the output about the state of the individual nations during any point of the Simulation	High
Req:08	US:09	Storeable state	The user has to be able to store the current state of the simulation, in order to be able to continue from it at a later point in time	Med
Req:09	US:08 & US:09	Active control	The user has to be able to pause the simulation during runtime, and to alter the parameters of nations while the simulation is running	Med-Low
Req:10	US:04 & US:06 & US:08 & US:11	Nation behaviour	The user has to be able to set limitations on how nations may interact to model various complex nation relationships	Med-High
Req:11	US:01 & US:09 & US:11	Deterministic RNG	The user needs to be able to rerun the simulation with the same seed and get the same results.	High-Med

2.2.2 Non-Functional Requirements

All concrete values presented in this section are merely initial suggestions before implementation of any prototypes, and before any testing on the actual implementation has begun, and are thus subject to change.

Label	Related User story	Name	description	importance
Req:12	US:01	Time Units	The simulation needs to run at a predefined (but as of yet undetermined) timescale, updating once per unit of time for each agent, and storing new information about the agents state, so it can later be presented	High
Req:13	US:07	Runtime	The program needs to take into consideration the runtime of the simulation, and ensure it does not become to long. The simulation should take no more than 1 hour for 1000 years to be simulated for 10 nations. This would mean that 1 year for 1 nation should take no more than 0.36 of a second.	Medium
Req:14	US:01 & US:03 & US:05 & US:10	Output format	The Program needs to produce output that is human and machine readable, and user friendly, producing both raw data, graphs, and text based output for the user	High
Req:15	US:01	User friendliness	The program needs to be intuitive to use, taking less than 3 hours to learn while unguided, and less than 1.5 hours while guided for a majority of users	Medium
Req:16	US:01 & US:07	Reliability	The program needs to be stable during runtime for typical usecases, not crashing outside of outside circumstances or extreme strain due to excessively taxing user input far beyond the expected norm	High
Req:17	US:01 & US:02 & US:03 & US:04 & US:06 & US:08 & US:09 & US:11	Initialization	The Program needs to allow users to easily input custom parameters, either via loading a file, via randomization with or without a predetermined seed, or by hand	High-Medium
Req:18	US:01	Ease of startup	The program needs to be easily run-able from the users desktop as a regular program, without having to download a simulation framework beforehand	Low

2.3 Risks & Mitigations

2.3.1 Risks

These are the risks currently identified to be relevant to the project, which will be considered for potential mitigation if determined to be sufficiently threatening to the overall success of the project.

Label	Related Requirements	Description	Likelihood	Severity
Risk:01	All	I might not be able to complete the project in the timeframe given	Med	High
Risk:02	Req:02	The Nations decision making is not realistic	High	High
Risk:03	Req:05 & Req:07	The output of the simulation is not user readable	Medium	High
Risk:04	Req:11	The Simulation is not Deterministic	Low	High
Risk:05	Req:18	The chosen framework does not allow me to provide the model to users that have not downloaded the framework	Medium	Low
Risk:06	Req:13	The program can not meet it's runtime requirements, and takes too long to complete	High	Medium
Risk:07	All	I face technical difficulties, and loose access to my workstation	Low	High
Risk:08	Req:15	The Program is not intuitive to use	High	Medium
Risk:09	Req:16	The program experiences frequent crashes during runtime	Low	High
Risk:10	Req:09	Framework does not support updating of parameters during runtime	Medium	High

		Likelihood		
		High	Medium	Low
Severity	High	Risk:02	Risk:01, Risk:03, Risk:10	Risk:04, Risk:07, Risk:09
	Medium	Risk:06,Risk:08		
	Low		Risk:05	

Figure 1: Risk Assesment

Plotting the severity of the damage the risk would cause to the project should it come to pass and the likeliness of the risk coming to pass into a grid, we get the results presented in Figure 1.
This shows that the most critical risks to be concerned about are:

- Critical:
  - Risk:02
- Severe:
  - Risk:01
  - Risk:03
  - Risk:06
  - Risk:08
  - Risk:10

2.3.2 Mitigation

The following are the proposed mitigations for the risks to the project's success determined to be at least "Severe".

Label	Related Risks	description
Mit:01	Risk:02	The entire success of the project hinges on the ability of the nations to make decisions. As such, extra time will be set aside during development to research how best to make nations' decision-making as close to realistic as possible.
Mit:02	Risk:01	To ensure that the project will be completed on time a full work plan has already been created and will be presented in section 3.1 and section B. Extra time has been allocated for most steps, to ensure that the project finishes on time or early.
Mit:03	Risk:03 & Risk:08	These risks can only really be ruled out with intensive user testing, which is currently outside of the scope of the project. I will however consider adding it if I find the time and resources over the course of the project
Mit:04	Risk:06	To ensure that the runtimes of the simulation are low, a powerful framework has been chosen that should guarantee the best possible results for runtimes. Additional time in the development process might also be used to further optimize runtimes, should they feel too sluggish.
Mit:05	Risk:10	This should be fixable via a workaround by instead storing the parameters of the simulation at the current time and then re-initiating the simulation with those parameters.

The other Risks were not considered for the case of this project as they were deemed too unlikely or too benign to spend much time on.

3 Methodology

3.1 Feasibility study

Based on papers discussed in section 1.4, it is clear that attempts to create global models have been fairly successful in the past, meaning that it is possible to create a model like the one I have in mind. The problem is how difficult it is to implement all of this functionality within a single semester, and with little to no background in simulation.
Thus, to see whether or not such an implementation would even be possible within the time frame I had available, I spent a significant portion of the semester researching a variety of Agent-based Modeling frameworks, that is, suites of tools that are intended to make it easier to implement your models.

This work is part of another course, and as such will not be discussed in depth in this paper, though the paper in question is available in the section A beneath in case the reader wishes to take a look at the work substantiating the decisions I am about to make. In the said paper, I conclude that out of the 5 popular frameworks that I studied, Netlogo[1] stood out as being of the utmost quality when it came to teaching new users how to use it, and for ease and speed of implementation of models once the user knew what they were doing.

Unfortunately, Netlogo is the lower end for computational power, and thus is ill-suited for taxing simulations, as well as making some assumptions about the agents in question which may be unnecessary for my project, and cause further slowdown.
However, Netlogo's quick and easy workflow has resulted in me choosing it as my framework of choice for rapid-prototyping of ideas throughout the project, as the program is easy to troubleshoot due to the sheer amount of online help available, and lends itself to quickly cobbling together a working solution so that it can be further refined in the main framework.

Which in this case is Repast[2], specifically Repast symphony, the java implementation of Repast. Repast, unlike Netlogo, is highly adapted to running computationally intensive simulations, and while in my experience its documentation is not as good as Netlogo's, it benefits of the fact that its modeling language Relogo is closely related to Netlogo's, so a lot of the documentation does carry over to an extend.

This only furthers to increase the translation from prototype to fully-fledged system, and hopefully, between these two systems, I will be able to implement the entirety of the system before the deadline.

3.2 Project Plan

I have created an initial outline for how I plan my workflow to be over the course of the spring semester, and how I plan to tackle the project. The below information can also be found in the Gantt chart provided in section B

3.2.1 Phase 1: Prototyping

Work on the project itself is slated to begin as soon as this paper has been delivered, and will run until the end of May when the Master thesis is due to be delivered.
Development of the project will begin with the creation of two prototypes, one developed over the course of a weekend via a high-implementation speed AMBS framework, and one developed over the course of another 2 days with a more powerful framework, the intent being that most costly mistakes are made in the framework that is quick to work with, and the results can then be achieved better in the new framework at comparable implementation times.

Once the low-fidelity Prototypes are finished, they will be used for the creation of a Report for the "Advanced project work report, finishing up the autumn semester. Work on the next phase of the project begins as soon as the semester ends, and before the start of the spring semester.

3.2.2 Phase 2: Research and Core implementation

During winter break I am scheduled to further familiarize myself with the chosen frameworks (as discussed in section 3.2), as well as use the initial prototype as a jumping-off point to start the core of the project, and test its functionality.

3.2.3 Phase 3: Cyclic development

From this point, project work is broken down into three identical cycles, each running for 6 weeks. During each of these cycles, I will be focusing on one of the key systems of the simulation, those being the Economy, International Politics and Warfare, and the Environment.
Each cycle will be broken down into a variety of tasks that are the same across all three of these systems.

- Research:
  As I am no expert on the fields of Economics, Politics, or Environmental factors, I will be spending the first workweek of each cycle researching the field that I am trying to implement, specifically the literature on what work has been done in simulating the field via agent-based simulation, and algorithms that govern how these factors impact the decision-making of nations.
  I have set off a little more than a workweek for each of these research blocks, as I figured I would need some extra time to figure out the fineries of such vastly different fields.

- Prototyping:
  Once again utilizing the same framework as previously, I will be doing a quick prototype of the system to mark down the rough functionality needed to make the system function. As the framework chosen for this job is rather fast to work with, I have placed down just under a workweek for this part of the cycle, but I can always use some of the extra time set aside for research for this part of the project as well.

- Implementation:
  Moving back to the main framework, I complete the implementation of the required systems and finish that part of the system as its own, self-contained unit with no outside influences. This does not include integration into the rest of the project, but merely represents the time spent creating the system as a standalone component.
  Since I will have already implemented the system as a prototype, I slate this part of the work to also take roughly 1 work-week, with a little extra time set aside on top of that in case I run into issues.

- Unit Testing:
  To assure that the system is working is attended, I will set aside a workweek for unit testing the systems, running partially concurrently with implementation once that is nearing completion, in case I do end up not needing the extra time set aside beyond the workweek, or just to be worked on in tandem with the main implementation.
  Should implementation take longer than expected, this workweek is another block of time I can eat into to increase the amount of time I have for implementation.

- Integration:
  Once internal validity of the system is ensured, I will begin work on integrating each system into the wider scope of the project, that is, to ensure that the input and output of the system take all other systems into account, so that the agents' decision-making is based on all three subsystems and their interplay. Nearly two work-weeks of time are set aside for this part of the cycle, but realistically will not entirely be used for that.

  As the earliest finished system will have a lot less work that needs to go into it for integration into just the core system, I assume that this amount of time will be overkill. In that sense, it acts as another buffer, should I fall behind pace in earlier parts of the project.

- Integration testing:
  Finally, the last part of the work cycle is integration testing, which is simply assuring the validity of all systems remains after they have been integrated with the newly finished system. A little over a workweek is set off for this, partially running concurrently with the Integration work itself, for the same reason as the unit tests.
  Again, this time will likely not be entirely used for the earlier projects, and as such acts as another buffer.

Once all three cycles have been completed, the implementation phase of the project is largely over. All in all, 20 weeks are set off for these three cycles, spanning the time between The start of January and the End of April. If everything goes as planned and no major roadblocks are encountered, I hope to finish the work earlier than that.

As I am almost guaranteed to not be so lucky as to smoothly sail through the entire project, I am using the extra time as buffers to help mitigate the risk of overshooting the deadline for the project.

3.2.4 Phase 4: Master Thesis

Finally, the month of May is entirely set aside for the creation of the master thesis, including an initial draft to be discussed with my supervisor, which should be done by the end of the second week of March. From that point forward, the rest of the time will be spent making the final version of the thesis, finishing up at the end of March, and being able to deliver my thesis on time before the 1st of June.
From that point on, all focus is on preparing to present the thesis.

3.3 Planned Contributions

Over the course of this Master thesis, I will produce a variety of different artifacts, each of which will also be a milestone towards finishing the project.
First of these is the Tool itself, initially simply named "Fictional History simulator", which, if I manage to realize the Project plan to its fullest, will simulate the economy, international politics, and environment of a set of Nations as defined at the start of the simulation, all according to the above specifications.

The overall system will be implemented as 4 distinct subsystems, all interacting to simulate the interplay between these different parts of a nation, and produce as realistic events as possible. Each of these systems will be developed independently, and as such will be largely decoupled, each system itself acting as a milestone towards the completion of the final project.
To the best of my knowledge, a tool that simulates vast timespans of historical data for nations of fictional nature and produces output that allows for the extraction of historical events of significance to work as a scaffolding for the creation of fictional settings does not currently exist.

It is currently still up in the air whether or not the final tool will be made publicly available, or whether it would remain a research artifact, purely for private usage. This largely depends on the results of the project, and to what extend the tool is complete by the time the project comes to an end.

Additionally, the Project will result in the creation of a Master thesis, which will be written at the project's end, detailing the work done, the results of the implementation, as well as potential future extensions, and any other relevant information about what work went into the creation of the system and the thesis itself.

Bibliography

[1] https://ccl.northwestern.edu/netlogo/. Accessed 08.12.2021.

[2] https://repast.github.io/. Accessed 08.12.2021.

[3] https://docs.google.com/spreadsheets/d/1JcX4sHAuBRGsbXIgktxj5n72sMyFQutQyqJ7R_xQCCU/edit#gid=0. Accessed 10.12.2021.

[4] The orr group industry report q1 2021. https://web.archive.org/web/20210812131438/https://blog.roll20.net/posts/the-orr-group-industry-report-q1-2021/, APRIL 2021.

[5] Timeline of arda. https://web.archive.org/web/20211004183453/https://lotr.fandom.com/wiki/Timeline_of_Arda, NOVEMBER 2021.

[6] Sameera Abar, Georgios K. Theodoropoulos, Pierre Lemarinier, and Gregory M.P. O'Hare. Agent based modelling and simulation tools: A review of the state-of-art software. *Computer Science Review*, 24:13–33, 2017.

[7] Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287, APRIL 2002.

[8] Richard W. Chadwick. Global modeling: Origins, assessment, and alternative futures. *Simulation & Gaming*, 31(1):50–73, 2000.

[9] Xu Zhang et Al. Improving cloud gaming experience through mobile edge computing. *IEEE Wireless Communications*, 26(4), AUGUST 2019.

[10] PAUL E. JOHNSON. Simulation modeling in political science. *American Behavioral Scientist*, 42(10):1509–1530, 1999.

[11] David P. Masad. Agents in conflict: Comparative agent-based modeling of international crises and conflicts. page 207, 2016. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2021-05-25.

[12] Thomas B. Pepinsky. From agents to outcomes: Simulation in international relations. *European Journal of International Relations*, 11(3):367–394, 2005.

[13] Adam Rowe. Traditional publishers are selling way more non-fiction than fiction. https://web.archive.org/web/20210226201130/https://www.forbes.com/sites/adamrowe1/2018/08/30/traditional-publishers-are-selling-way-more-non-fiction-than-fiction/?sh=1b27dbda56d0, AUGUST 2018.

[14] Leigh Tesfatsion. Agent-based computational economics: modeling economies as complex adaptive systems. *Information Sciences*, 149(4):262–268, 2003.

A APPENDIX A; Report for IMT4134: Specialization in Software Engineering

Please note that this is not part of the project
it merely serves as a ressource, incase the reader wants to learn how I came to the conclusion to use Netlogo and Repast as my frameworks of choice for the project. **AGAIN. This is not part of the Report itself, is not work I did for this course, and most certainly is not me double dipping to receive a better grade in this course. PLEASE ignore everything in this appendix for the purposes of grading my paper.** It is merely here to substantiate my choice of frameworks, incase anybody cares to look.

Specialization in Software Engineering

Leon Cinquemani

December 2021

Abstract

Agent-based modeling is a sub-discipline of Simulation which focuses on modeling systems from the bottom up, as a series of interactions between agents, each of which has a series of potential actions it can take, and a series of circumstances in which it takes those actions.
Agent-based modeling is commonly used across a variety of disciplines, from flow management to social sciences, traffic-control, and organizational protocol development. However, scientists specializing in these fields might not necessarily have the programming know-how to create a model from scratch.

This is where Frameworks come into play, simplifying the process of creating models by providing a suite of tools to define agents, the actions they can take, and the metrics which need to be collected from the simulation.
Finding the right framework for the task can be challenging, as a large variety of different frameworks of varying prevalence are available, both freely available to the general public and as proprietary software.

This paper focuses on a small subset of Frameworks for Agent-Based Modeling and Simulations, evaluated by a variety of qualitative, subjective metrics, based on the author's experiences with each of the tested frameworks, and hopes to give insights into the perceived shortcomings of the different frameworks.

Contents

1 Introduction

1.1 Background

The field of Agent-based modeling is fairly wide, the concept of building a system from the ground up having found use in a variety of different fields, from social sciences to flow management.[1] As such, interest in the field has only been increasing since its inception, as more and more fields become aware of the benefits of modeling systems from the bottom up.

Agent-based modeling focuses on creating a system comprised of agents, each of which carries data and methods, and interacts with the environment it has been placed in, as well as other agents present. This allows Agent-Based Modelling and Simulation (ABMS) to represent complex interactions between a multitude of actors in a natural way, and to provide insights into phenomena that rely on the interactions of individuals, which might be difficult to accurately model mathematically.

With the applicability of agent-based modeling being very broad[1], and interest in the field expanding, it is important to provide researchers the tools they need to create their own models, adapted to the specific issues they are researching. For this reason, a large variety of ABMS Frameworks exist, which focus on making the creation of agent-based models as easy as possible, while at the same time trying to (generally) remain universally applicable for all potential use-cases of ABMS.

Since Agent-based modeling and simulation is a mature tradition of simulation by this point, there are a lot of varying frameworks that attempt to provide this assistance to intrepid researchers, of varying levels of complexity and specialization[2].
This does mean that researchers might be initially overwhelmed by the number of options available, as the right tool for the job might not immediately be clear. Studies, such as the one conducted by Abar et Al[2], seek to classify and simplify the field by sorting the available frameworks via a variety of criteria. Others, such as this paper and the study by Railsback et Al[3], focus on a small subset of these frameworks, and how they compare against one another in a variety of metrics.

1.2 Definitions

An ABMS framework, put simply, provides a set of standards and software tools[3] for developers to create their own implementations of simulation models, which can then be run and analyzed to gather insight into the underlying rules of agent interactions.
For the sake of this paper, I will be considering both frameworks such as NetL-

ogo, which come with an integrated IDE, and frameworks such as GAMA and Cultsim which come without their own IDE but can also be used with an IDE to write their models.

The test model I will be implementing in each framework is the classical, simple boid model, that is, a flocking behavior simulation in which each agent attempts to:

1. Keep a certain minimum distance from every other agent
2. Orient it's heading to be equal to the average heading of its neighboring agents
3. Maneuver in such a way as to be as close as possible to the center-point of its neighbors' positions

Each agent moves in 2D space at a constant maximum speed and can turn a specific number of degrees each turn to correct its heading based on the aforementioned criteria.

1.3 Research Questions

The reason for me choosing to undertake this project is already substantiated above, as the applicability of Agent-based models and the need to find a functional, robust framework to work with necessitate the researching of the differences between these ABM platforms to allow researchers and hobbyists to make informed decisions when choosing which framework to use when developing their models.

For my project specifically, I also had the incentive that the overarching project of my master thesis would be the implementation of a model, which meant that I found myself in a situation where I needed to choose a framework from the list of available options. As such the project also has a pragmatic reason for me. Overall the Research questions I seek to answer over the course of this project are:

1. Which Frameworks are the most user-friendly in terms of availability of online help and quality of documentation?
2. How long does it take to implement a pre-determined model for somebody that has little or no experience with the framework.

To get a broad overview of these issues, I chose a variety of Frameworks, originally examining a total of 5, with 4 of these ending up being tested, and 3 producing a full implementation of the Boids model. This was due to issues with some of the frameworks tested either providing insufficient information to successfully operate them, or being too poorly maintained to be effectively used.

2 Methodology

2.1 Finding Frameworks

For the process of finding frameworks for this project, I drew initial inspiration from a Wikipedia article on ABMS frameworks[4], where I first picked up MASON, Netlogo, Repast, and GAMA. These choices were further refined and confirmed after looking over the classifications provided in the paper by Abar et Al[2], where the previously mentioned Frameworks were considered to be capable of running models of medium to high intensity, and varied widely in how much effort went into creating a model, with Netlogo being considered the easiest to work with, and MASON and Repast both being considered difficult to model in.

These 4 frameworks were chosen based on several criteria:

- The Framework must be available to the public free of charge
- The Framework must be fully available on a Windows-based device
- Development and maintenance on the framework must still be active

I deemed these as reasonable requirements for a Framework to be used by a novice modeler, as they ensure a low barrier of entry, with the average beginner likely unwilling to spend money on something they have no experience in, and with Microsoft being used by 80+% of all personal computers.
The final criteria were added to ensure that the framework would be decently modern, following current best practices and utilizing modern technologies, to thus provide a contemporary look at the state of AMBS frameworks.
These constraints also helped choose a specific version of the Repast framework, as Repast Symphony was the only version available that satisfied all three criteria.

Additionally to these 4 frameworks, I decided to also include an in-house ABMS framework, developed by myself and a pair of fellow students, named Cultsim[5]. Cultsim was developed to allow for high-quality visualization of simulations and made to be platform-independent.
It was included both as a potential candidate for future work for myself, and as it fits most of the criteria mentioned above, except for a lack of maintenance since its initial completion.

The final 5 chosen frameworks were thus:

- Cultsim[5]
- MASON[6]
- Netlogo[7]

- Repast Symphony[8]
- GAMA[9]

2.2 Testing

To measure the frameworks, I looked at several metrics that I deemed to be significant based on the above research questions, some of which were easily quantifiable and some were not. Due to the nature of these tests, they ultimately are partially subjective, as a number of metrics are based on my personal experiences with the frameworks in question.

For this paper, I focused on the following criteria:

- Implementation
  - Time taken to implement boids when never having used the framework before
  - Time taken to re-implement boids after having done so before
- Documentation
  - How easily available online documentation for the framework is
  - The quality of the documentation based on:
    - Examples of use
    - Ease of navigation
- Assistance
  - Number of related questions on Stack-overflow
  - Number of Google search-results
  - Number of video-results on Youtube
  - Number of Papers citing the framework

2.2.1 Implementation

For the purposes of Implementation, I will be creating a model for Boids, as discussed above, from scratch for each of the frameworks. This implementation will be completed twice, and the time taken for both attempts will be recorded. Ideally, the first implementation will show how long it takes to implement a simple model with no background in the framework, and the second will give insights into how long it takes to implement the same model when the researcher knows how to do it. This should help provide an overview of how easy it is to work with the framework once one is familiar with how it functions.

There are some issues with this, however, as skills learned while working with one framework might translate over to the next, thus meaning that the

tests are not done entirely in a vacuum, as they ideally would be. As I am but one researcher with limited time to complete these tests, I have found no way of preventing this from occurring. Therefore, I will present the test results in order, and highlight any potential influences on implementation time from sources other than simply learning the framework.

2.2.2 Documentation

Documentation includes all official documentation that comes with the software, both installation instructions, step-by-step implementation guides, and reference manuals.
These will be rated based on how easy they are to find, how clear they are (how well the functionality of different parts of the framework is explained), and how easy it is to find documentation for individual primitives and functions that may be relevant to the model.

I would also report on the completeness of the documentation, as to say how much of the framework's functionality is documented, but that would require a deeper understanding of the frameworks than I possess, seeing as this study is my first interaction with most of these systems. As such, I cannot say if any parts of the system have been left out of the documentation unless it is blatantly obvious.

2.2.3 Assistance

In this category I will discuss how easy it is to find help for the framework in question from non-primary sources, that is sources outside of FAQs and documentation provided by the developers. For this study, I quantify this metric as the number of solved problems on Openstack that can be found by searching for relevant keywords, the number of google searches that come up when looking for the framework, and the number of papers published about the framework in question (if available on the developers' site).

I hope that this gives a fairly detailed overview of non-primary sources of information that can be found about the framework, and that would be intuitive for novices in the field to look at.

3 Results

3.1 Installation

To utilize these frameworks, step 1 is obviously to install the framework. This task is more involved for some of them than others, and one of the frameworks

managed to get itself removed from further testing at this step. In cases where it was required, I looked online for assistance with installing the frameworks, both from official and unofficial sources.

3.1.1 Netlogo

Netlogo was easy to install, as the installer contained everything required to run the program from the get-go. As such, installing the framework required no further assistance of any kind. The web page is a little confusing, as it contains fields to enter one's name, organization email, and any comments one may have, but these fields are entirely optional for downloading, it would seem.

The website also allows users to choose a specific version of Netlogo, but as I have no requirements for versioning, I left it on the default selected value. After downloading the installer, I only needed to choose an installation folder, and the program was installed and working.

Out of all programs surveyed, Netlogo was the easiest to get running, and the other frameworks definitely could benefit from making attempts at a similar user experience.

3.1.2 Gama-platform

Gama was roughly on the same level as Netlogo when it came to its installation, only requiring the user to choose an appropriate version. As a Java illiterate I chose the version with a built-in Java Development Kit so I did not have to worry about it, and then download the latest version of the framework.

From there, I received a .zip file that could simply be unpacked in a location of my choice and could be run from there, problem-free. Not as clean as a dedicated installer, but still very easy.

3.1.3 Repast Simphony

Repast also was simple to install via an installer, which was very appreciated. An installer is available for download from their website, which upon opening easily walks users through the steps of setting up the program, including installation destination, and whether to include a dedicated installation of the JDK, which I checked "yes" on.

The program otherwise requires little to no setup. Overall, another very smooth installation experience, slightly lessened by the fact that the installation only seems to run when booted from the start menu, for hitherto unknown

reasons. Additionally, the installation present on my machines seems to randomly decide that my work-space is corrupted, and purging the files present in it. I blame this for the lack of implementation-code available for Repast in the Appendix, and sincerely apologize.

3.1.4 MASON

Where the other publicly available frameworks were easy to install either via a simple .zip file download or via an installer, I could not get a functioning version of MASON to install itself on any machine I attempted it on. The download site presents a variety of files to download, from a .jar file which supposedly functions as a binary distribution, to a full source distribution.

According to the site, the latter of these two is the common installation, so I chose that one (though I attempted using both). This presents me with several files, including a Readme, which I next consult for what to do. It speaks of a "jar" folder, in which the file with which to run the program should be located, but the distribution does not seem to contain such a folder unless it is buried deep within the folder structure, which seems as though it would be a rather strange design choice.

The file also speaks at length of adding libraries to your Classpath, but as somebody that has little to no know-how of how Java works, it becomes clear to me that while I can follow tutorials online there is little to no way for me to see if what I did actually worked.

Likewise, the Manual for MASON, provided on their website is rather unhelpful and serves seemingly to remind me that I am not the intended audience, as it repeatedly refers me to different frameworks, namely Netlogo and Repast. I spent a good few hours trying to troubleshoot the problem to no avail, and am forced to remove MASON from the list of frameworks I am testing from this point forward, as my know-how with Java applications is seemingly not good enough to even install the program.

3.1.5 Cultsim

Cultsim is a similarly sad tale to MASON when it comes to user convenience when downloading and installing the file. The entire project is available via a GitHub link[5] and can either be cloned as a source-code project or be downloaded as a set of binaries. I tried both and found the latter to be far easier to get working, as the source distribution struggles from poor maintenance and several dependencies have since been updated to include errors that break the project.

Via the binaries, I receive a simple .zip file, and after unpacking am greeted with a ".exe" file, as well as a few others, including a data.zip file. Unzipping that as well makes the program functional, and it runs just as the source distribution does, albeit with less of a headache to get it up and running.

Overall, the binary installation was entirely pain-free up until this point. The installation instructions on Github were also concise and informed me of what to do to get the program running step by step. I would still say it is the most convoluted to get set up before running out of the four frameworks that I managed to install, but by comparison to MASON, its installation is relatively painless.

Installation alone has already removed one of the 5 frameworks from further testing, which means that from this point forwards, MASON will no longer be considered for the rest of the study. Still, I found it valuable to include an example of a poorly realized, convoluted, and confusing installation process, as I could not get the program to correctly install, even by searching for online assistance for installation. This, coupled with the less than helpful user manual and Read me make me feel justified in dropping the framework at this point, as I doubt that somebody new to ABMS frameworks would continue troubleshooting, and rather swap to another option.

3.2 Boids

From this point, I continued with the 4 remaining frameworks and proceeded to attempt to implement the Boids framework in each of them. While the exact implementation varied (Complete code for each implementation of Boids available in the appendix below), the pseudocode could be described as follows:

```
//step 1: Setup
Reset world
Create x agents with x = random & y = random

//step 2: Per tick
For every agent:
    update_direction()
    move()

//update_direction
neighbours = other agents within radius
if(distance to closest agent < minimum distance)
separate()
else
alignment()
cohesion()
```

```
//separate
direction = direction towards closest neighbour + 180
return direction

//alignment
heading = heading of each neighbour
heading = heading / neighbours.size
direction = towards heading
return direction

//cohesion
x = average x coordinate of neighbours
y = average y coordinate of neighbours
direction = towards [x,y]
return direction

//move
set heading = direction
move_forwards(speed)
```

The implementation varies from Framework to framework, but overall stays true to this formula.
As mentioned previously this test was implemented twice in each framework, in the hopes of capturing both time required to learn the framework, as well as how long it would take to implement once one is familiar with the said framework. To expose any potential biases towards having an easier time learning later frameworks due to experience gained from earlier frameworks, these results will be presented in order of implementation, and the results will be discussed below.

3.2.1 Cultsim

As can be noted in figure 1, Cultsim does not have any data associated with it. This is because I found myself unable to implement Boids as per the above specifications into Cultsim as is, as I would have to modify the source code of the framework to allow for finding the distance and direction to the closest neighbor, and thus am unable to implement separation.

The example implementation of Boids within Cultsim also does not include separation, and thus I am unable to fully implement Boids to the specifications set. Additionally, even when I attempted to implement the less complete version of Boids without separation, my code refused to run, even when I created a near 1-1 copy of the Boids code supplied with the framework. Neither the version created for the binaries nor the version in the source distribution managed to run without crashing.

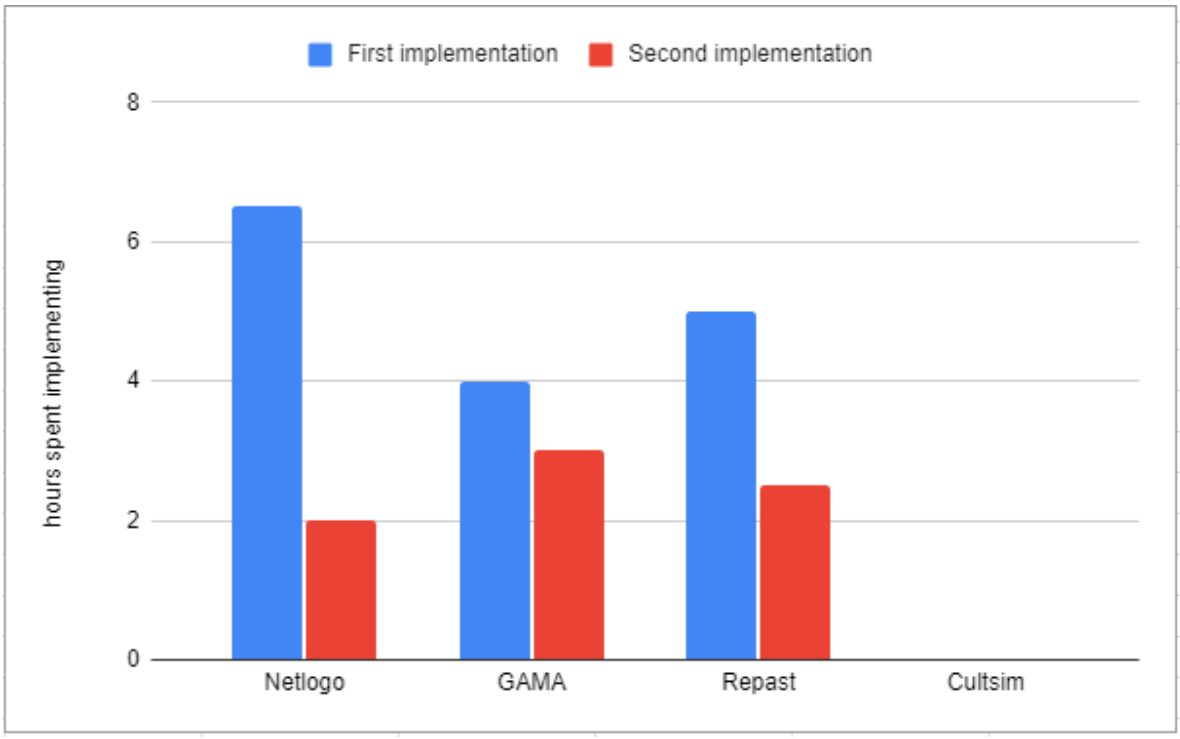


Figure 1: Time to implement Boids in Hours

I found no easy fix for the issue in the (unavailable) documentation and therefore was forced to abandon Cultsim for this test at this point. The binary version did not seem to allow for users to create their own scenarios or required additional steps not mentioned in the documentation and unknown to even me, one of its creators.
The source code version did allow me to implement my own scenarios after I resolved most of the dependency errors, but continued to crash, even though the source code was near identical to the code of the example implementation.

I, therefore, deemed it unlikely that somebody with no further knowledge on AMBS frameworks or Cultsim would attempt to troubleshoot much further, especially given the seemingly broken documentation of Cultsim, which does not build when following instructions from the source distribution, and is not included in the binary distribution, which seems like another oversight.

3.2.2 Netlogo

Netlogo was touted in every source I came across as the most beginner-friendly of the frameworks, and thus I decided it would be a good starting point for the project. The first attempt at implementing Boids cost me roughly 4 hours and did not end up functioning as intended. This was due to several logical errors made in the code of the model itself, which ended up taking a lot of time to fix.

Luckily, it was at this point that I noticed the example projects that come included with Netlogo, the likeness of which are also included in each of the other frameworks that I tested. This made the implementation of the code a lot easier overall as I now had an example to look at for how to do the more complex parts of the Boids implementation and the syntax for the framework. This drastically sped up working times, and I managed to finish the implementation of the model within an additional 2.5 hours.

This time was drastically reduced for the second attempt, which ended up merely taking 2 hours from start to finish, the shortest time for re-implementation noted during the tests, producing an equivalent implementation in less than 1/3 of the time it took to originally create the Model.

3.2.3 GAMA

The Gama implementation of the Project proved to be relatively painful to complete, mostly due to the framework's documentation lacking several key concepts that I would have expected from a framework of this sort, such as the concept of a vector, vector normalization, and similar concepts. The example implementation of Boids also proved to be rather crude, as it did, in fact, **not** normalize its vectors, instead using several multipliers to allow users to set how heavily separation, alignment, and cohesion are weighted.

While one might say that this allows users to steer the importance of the individual components of the flocking behavior, I would argue that the same level of interactivity could have been achieved better by first normalizing the vectors, and then multiplying them with a multiplier between 0 and 1.
That at least is my opinion on the matter, others may disagree. As user interactivity was not a part of the pseudo-code requirements, I did not bother to implement it.

The second time around still took an inordinate amount of time, as the way Gama handles its model implementations is rather cryptic and the documentation is, again, rather poor. Overall, the time taken only went down from 4 hours for the initial implementation to 3 hours for the second attempt. This makes GAMA the slowest framework when it comes to re-implementation.

3.2.4 Repast Simphony

Repast is a very complex program to set up, and documentation for getting the program up and running is rather sparse, with the tutorials being a little more cryptic than what would be appreciated. Online video tutorials for Repast were largely nonexistent, and the written setup tutorial had screenshots that did not apply to my installation of the program, making finding the correct buttons for setting up the project somewhat difficult, especially as a ReLogo Project and a Repast Project look very similar if one is not looking carefully.

With the initial hurdles overcome, however, Repast proves itself to be a rather straightforwards framework, aided by the fact that ReLogo, the programming language used by Repast, is closely related to the language employed by Netlogo, and thus a lot of syntax and functionality carried over. This made it easy to implement the test once I understood that coding for Repast, in the simplest sense, was a lot like coding for Netlogo in java.

This drastically sped up the process of implementation, and the second test only took a little longer than implementing in Netlogo, despite the additional overhead from having to implement visualization, which Netlogo does automatically.

3.3 Documentation

3.3.1 Netlogo

Netlogo's documentation is easily the best of the four, containing a complete dictionary of all concepts used by the frameworks programming language, including several tutorials, and easily being available in a variety of formats from the front page under the user-manuals section.

Netlogo's documentation provides step-by-step tutorials for implementing some simple models, guides for all key parts of the creation of a model in Netlogo, and a dictionary of all concepts, primitives, and functions available to creators in Netlogo, complete with examples of usage for the non-self-explanatory concepts.

Netlogo's documentation is as good as I have seen documentation get for any software I have worked with and stands head and shoulders over the documentation provided by its peers. For the entirely programming illiterate, I can only assume that this alone would set Netlogo far and above the rest, as it seems to me that one could create models for Netlogo simply by looking over the documentation and using the dictionary provided.

3.3.2 GAMA

While extensive, I find myself unable to praise GAMA in the same way that I praised Netlogo. There is nothing particularly wrong with the documentation, as it provides much of the same information that Netlogo provides, yet I feel that it is presented in a more cluttered, less transparent way, perhaps because each section must be opened into a drop-down menu before it's contents are shown. The way operators are handled is also less than appealing in GAMA, with them being sorted alphabetically, rather than being linked to the individual data structures they operate on.

The way that the complete dictionary is presented in GAMA also is nowhere near as intuitive as Netlogo, and I frequently found myself having to ctrl+f my way through it when attempting to find information on a specific function and variable.
The documentation feels cluttered and is harder to navigate than that of Netlogo, but it is very much serviceable overall, even if it gives me the feeling that I am missing key parts of it because I do not know how to find them.

Most concepts are explained well enough, with a majority having a small example of usage next to them, barring a few exceptions. Overall, the documentation is not as good as that of Netlogo, but still decent enough.

3.3.3 Repast Simphony

Repast's documentation is neatly divided into a lot of different files, some more easily noticeable than others. The complete reference guide seems rather offputtingly designed again and does not include a reference for functions and similar primitives provided by Repast, but it does come with detailed explanations and screenshots for most of, if not all the key concepts of Repast.

The actual list of primitives and associated functions can be found in the Relogo primitives file, which has most of what I would want from a list of available functions and primitives, but suffers due to the fact it scrolls both vertically *and horizontally*, making the page hard to navigate.

Other than these gripes, the tutorials are well implemented and decently well-annotated, giving users a sense of direction as to where to start when learning the program, and how to progress forward through learning the framework. Overall it is serviceable and less cluttered than the Gama documentation.

3.3.4 Cultsim

Despite the assurances on the main page that Cultsim allows users to build the documentation after downloading the source code version, I have been unable

to get this functionality to work properly, creating no useful documents whatsoever.

What is worse, the binary implementation does not come with any form of documentation, and no help is available online.
Ironically, even if I managed to get the documentation to build, that would not help me in implementing any models, as the documentation for the Lua scripting, the part of Cultsim that is used for model definition is unfinished, and thus unavailable.
Meaning, that even if I had managed to build the documentation, that would only help me with changing the source code, not actually writing a model.

Thus, I once more have to give Cultsim a failing grade in this study, as the program does not provide any valuable documentation to the user beyond the example implementations present in the files, making it archaic and non-user-friendly, even if it were to function as intended.

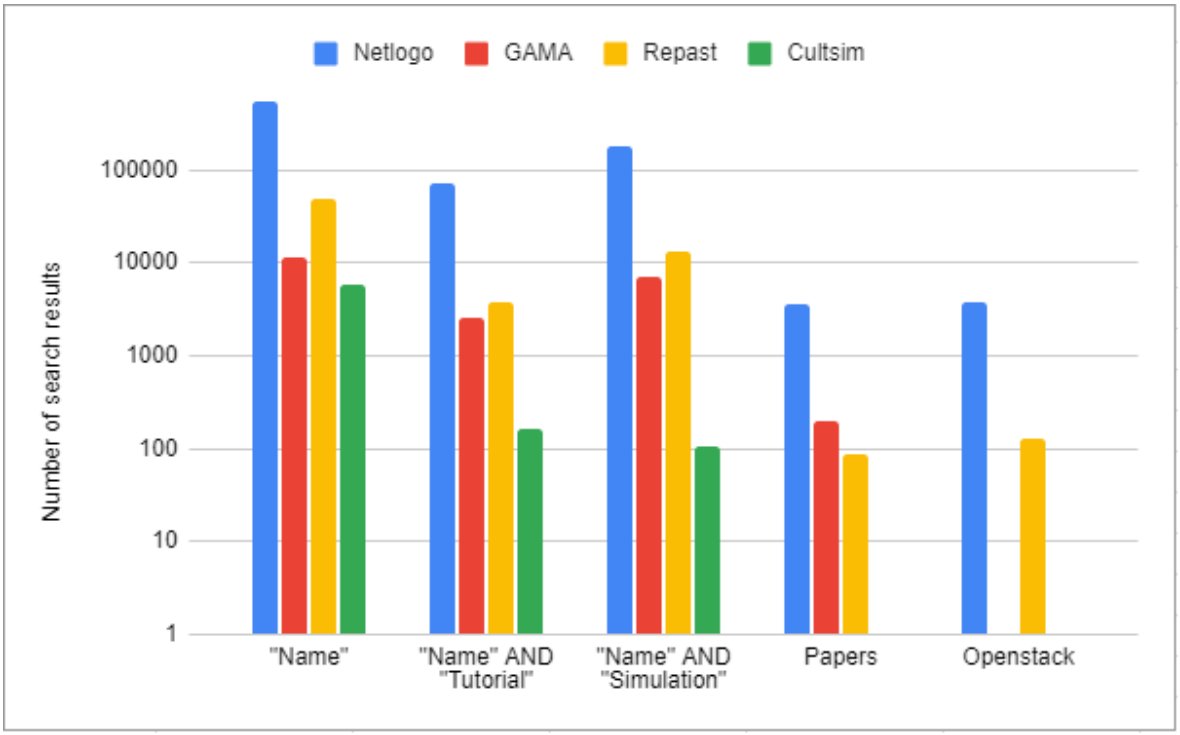


Figure 2: Search results across various sites

3.4 Assistance

In this section I will be elaborating on the amount of online help that could be found for the individual frameworks via the metrics previously defined as presented by the above figure 2, these being:

- Number of Google results
  These are separated into three different categories by the search terms used to find the results. These are "Name of Framework", "Name of Framework" AND "Tutorial", and "Name of Framework" AND "Simulation", as these seem like the most likely search queries one would use to find information and help for these frameworks specifically. The requirement of including the exact phrases requested should lower the number of false positives, that is irrelevant results being counted.
  This can however not be eliminated, as shown by the fact that Cultsim has several thousand search results, despite only having one relevant result, that being the actual GitHub page from which one can download the framework.

  Still, as I am including the irrelevant results for the other frameworks, it would feel unfair not to do the same for Cultsim. In this capacity, it also serves as a rough estimate on how many false results might be present in the searches for the other frameworks.

- Number of academic papers
  This is the number of academic papers associated with the frameworks, as denoted by the papers listed on the websites of the frameworks. This is by no means an exclusive list but should give some idea of which frameworks are most cited in academic works.
  It should also be noted that technically Cultsim does have 1 paper written about it, that being the bachelor thesis that it was created for, but that paper is not available on the Github-page that serves as Cultsim's homepage, and thus is not included, leaving its number of academical papers as 0.

- Number of questions on Stack overflow This includes both open and resolved questions, as found via the tag [Framework name]. The number is reported by StackOverflow's search results. Surprisingly StackOverflow has no active tag for the Gama platform, and no relevant results come up when searching for it in any way I have found. Thus, it shares last place with Cultsim, having no questions of Stack-overflow.

4 Discussion

4.1 Issues

4.1.1 Subjectivity of tests

Due to the less than objective nature of some of the tests, the reproducibility of my results, particularly in the fields of time required to learn the tests and the ratings of the documentation, is likely to not be very high.
Ideally, these tests would have included results from more test subjects, to get a more generalized overview of the metrics mentioned. However, I did not find myself in a position to include more testers due to concerns regarding both privacy, as well as public health and safety due to the Covid-19 pandemic currently ongoing, and the resulting lack of easy access to test subjects.

Still, I believe that the overall trends noted, and general commentary on the individual framework in every category tested has merit, and might aid in building an understanding of how the 5 frameworks discussed stack up to one another, both in terms of how usable they are, and how easy they are to pick up initially.

While by no means a definitive answer, I believe that the data provided still presents the experiences that I had when attempting to learn these 5 frameworks faithfully, and hopefully can be of assistance in choosing among these frameworks.

4.1.2 Inaccuracy of results

Due to a variety of factors, most of which I already touched on, several of the test results are likely somewhat skewed. The outside variables noted will be presented here once more in more clarity to allow for a more critical overview of what the numbers presented mean.

- Increased familiarity with the field and frameworks
  As mentioned previously, it is likely, that my methodology for researching the frameworks and implementing my test model improved over the course of the study, thus skewing the results in time taken to implement the tests in favor of the frameworks that came last.

  I think this is most clear in the time taken to originally implement Boids for Netlogo, which was before I found the example models included with each of the frameworks, which were heavily used in the following implementation work. This, in addition to me growing more familiar with how to implement Boids over time likely inflated the time taken to originally

implement the test for Netlogo significantly.

If the first, failed attempt at implementing Boids for Netlogo had not happened, I would think that the times for implementing Boids in Netlogo the first and second time would have been near-identical, as the implementing of simple models is very easy in Netlogo, as the time to re-implement the model the second time prooves.

I also think it should be noted that Repast's model was made significantly easier to implement once I realized that a lot of Netlogo's excellent documentation near-perfectly translated for ReLogo, and thus it is likely that I was able to finish the initial implementation faster because of this, and might have taken longer if I had gone through the Frameworks in the opposite order.

- False Positives for Cultsim Searches
  Unfortunately, there exists a videogame by the name of "Cultist simulator"[10], often shortened to Cultsim by its fanbase when discussing it on the internet. This means that all three search results for "Cultsim", "Cultsim" AND "Tutorial", and "Cultsim" AND "Simulation" on google are heavily inflated with results about this game, tutorials for playing the game, and the fact it is a simulator.

  There is not much that can be done to eliminate these results from the list and keep within the boundaries of the predefined search keywords. The best I have found is the adding of the "-game" keyword to the google search, slashing the results down to a third of their original number. However, including the same keyword for "Netlogo", that is "Netlogo" -games somehow returns more results than just "Netlogo".

  As such, I have decided to leave the values of Cultsim as is, and clearly stated in the paragraphs pertaining to Cultsim's search results that they are heavily inflated, in hopes of not only providing an overview of how inaccurate google results for the other frameworks might be but also to hopefully not sabotage the results of those searches with the strange quirks of the google search bar.

  I will state it again here, to make certain it is noted. The accurate number of websites that contain information on the framework Cultsim is less than 100, and likely less than 10. The heavily inflated results are included in the list merely as a measuring stick for potential inaccuracy in the other search results and as a warning against trusting google search results to be an accurate metric.

4.2 Future work

As for future work in the field, I have a few suggestions on how to continue on the work presented in this paper, as well as a few ideas that were originally meant to be part of the work done for the here presented study, but that ended up having to be cut due to time constraints.

4.2.1 More Frameworks

The first and most obvious continuation of the work here presented would be the application of the same or similar tests to a wider range of Frameworks. Ideally, at least to the point where one could get Mason to work.
Having more frameworks tested and presented would result in a better overview of the field, and thus serve as a more valuable resource to future researchers. Additionally, it would help to further highlight trends in the world of ABMS frameworks and their components, such as the ease of installation, quality of documentation, and the impact it has on how easily adopted the individual frameworks are. Seeing as Netlogo had good scores across the board, ignoring the inflated time it took to get the first implementation done, I wonder if that is specific to Netlogo, or if other characteristics can explain the outstanding performance of Netlogo, such as its integrated IDE, or it's automatic visualization, for example.

4.2.2 More Tests

Making more models to test implementation times for the frameworks would result in a richer, more conclusive set of results, and help to eliminate some of the biases from becoming accustomed to the possible issues that need to be faced when implementing the model across different frameworks.
Perhaps changing up the order in which implementations are performed for each new model would help in further alleviating these biases.

Originally I had planned to supplement the "Boids" model with models for "Food scavenging" and "Predator avoidance". However, both of these needed to be cut due to time constraints. As fairly simple problems, they seem as though they have example models in each of the frameworks tested (though those in Cultsim might not be entirely functional), and as such should make for good candidates for future tests.

4.2.3 More Testers

To reduce the subjectivity of the tests that deal with the time taken to implement the test models, as well as the metrics for how good a framework's documentation actually is, I find it likely that increasing the number of people

running these tests would be beneficial.
This does however require both the recruiting of more testers that are willing and capable of implementing these models, as well as the consideration of privacy and health concerns, all of which were far beyond the scope of this study.

5 Conclusion

Over the course of this paper, I have presented the results of roughly 3 months of studies on a variety of agent-based modeling and simulation frameworks, and implementation of a simple Boids model for each of them to study how long it takes to learn the framework, as well as how long it takes to implement the model a second time once one is familiar with the framework.

I have gone into detail on my findings on the documentation, tutorials, and outside assistance provided for each of the frameworks presented, excluding MASON, which had to be removed from further testing early due to me being unable to successfully install it on my machine by following installation instructions to the best of my ability.

I have presented my results for each of these tests, as well as discussed what could have been done better during the project, and what could be done in the future to expand on the work that I have presented.

Overall, the results point towards Netlogo and Repast being the two frameworks that are the most user-friendly, being easy to install, having good or serviceable documentation, and not taking long to model with once one has learned how they work. Additionally, I will highlight Netlogo's quick workflow, which allows for the framework to be used for rapid prototyping, which is another point in its favor. It is up to the task at hand whether the expanded toolset of Repast is worth the additional effort in implementation compared to Netlogo.

MASON and Cultsim are the least user-friendly of the frameworks, the former being difficult to install, to the point that it had to be excluded from the remainder of the study, and the latter being poorly maintained, to the point that several dependencies required for the program to run are broken.
This necessitates a lot of work from users to even get the framework up and running for both of them, and thus is not advisable for beginners or non-programmers.

I will however highlight here that out of all the Frameworks, Cultsims visual representations were of the highest quality, which was the highlighted feature of the framework. This does however not counterbalance the glaring flaws highlighted throughout the study, but I still found it worth noting.

References

[1] Eric Bonabeau. "Agent-based modeling: Methods and techniques for simulating human systems". In: *Proceedings of the National Academy of Sciences* 99.suppl 3 (2002), pp. 7280–7287. ISSN: 0027-8424. DOI: 10.1073/pnas.082080899. eprint: https://www.pnas.org/content/99/suppl_3/7280.full.pdf. URL: https://www.pnas.org/content/99/suppl_3/7280.

[2] Sameera Abar et al. "Agent Based Modelling and Simulation tools: A review of the state-of-art software". In: *Computer Science Review* 24 (2017), pp. 13–33. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2017.03.001. URL: https://www.sciencedirect.com/science/article/pii/S1574013716301198.

[3] Steven F. Railsback, Steven L. Lytinen, and Stephen K. Jackson. "Agent-based Simulation Platforms: Review and Development Recommendations". In: *SIMULATION* 82.9 (2006), pp. 609–623. DOI: 10.1177/0037549706073695. eprint: https://doi.org/10.1177/0037549706073695. URL: https://doi.org/10.1177/0037549706073695.

[4] https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software. Accessed 08.12.2021.

[5] https://github.com/Togtja/CultSim. Accessed 08.12.2021.

[6] https://cs.gmu.edu/~eclab/projects/mason/. Accessed 08.12.2021.

[7] https://ccl.northwestern.edu/netlogo/. Accessed 08.12.2021.

[8] https://repast.github.io/. Accessed 08.12.2021.

[9] https://gama-platform.org/. Accessed 08.12.2021.

[10] https://weatherfactory.biz/cultist-simulator/. Accessed 09.12.2021.

6 Appendix

6.1 Netlogo code

6.1.1 Implementation 1

```
to setup
 ca
  reset-ticks
  create-turtles num-turtles[
    setxy random-xcor random-ycor
  ]
end

to delete
  ca
```

```
  reset-ticks
end

;;Get The heading of the turtle towards the average heading of n neighbours
to align [neighbours]
  let xdir sum [dx] of neighbours
  let ydir sum [dy] of neighbours
  if xdir != 0 and ydir != 0
  [turn subtract-headings (atan xdir ydir) heading]
end

;; Get the direction towards the average position of n neighbours
to cohesion[neighbours]
  let xdir mean [sin (towards myself + 180 )] of neighbours
  let ydir mean [cos (towards myself + 180)] of neighbours
  if xdir != 0 and ydir != 0
  [turn subtract-headings (atan xdir ydir) heading]

end

to avoidance
  let neighbour min-one-of other turtles [distance myself]
  turn subtract-headings heading ([heading] of neighbour)
end


to boids[neighbours]
  ifelse (distance (min-one-of other turtles [distance myself])) < min-dist
  [avoidance]
  [align neighbours
    cohesion neighbours]
end

to turn [dir]
  ifelse dir > max-turn
  [ifelse dir > 0
    [ rt max-turn ]
    [ lt max-turn]
  ]
  [ rt dir]
end

to go
  tick
  ask turtles[
    let closest-turtles min-n-of num-neighbours (other turtles) [distance myself]
```

```
   boids closest-turtles
   forward 1
  ]
end
```

6.1.2 Implementation 2

```
turtles-own [ neighbours nearest-neighbour]


to setup
ca
  reset-ticks
  create-turtles num-turtles[
    set xcor random-xcor
    set ycor random-ycor
  ]
end

to go
  tick
  ask turtles[
  boids-heading
  forward 1
  ]
end

to clear
ca
reset-ticks
end

to boids-heading
  set neighbours other turtles in-radius radius
  if(count neighbours > 0)
  [
    set nearest-neighbour min-one-of neighbours [distance myself]
    ifelse(distance nearest-neighbour < min-dist)
    [avoidance]
    [cohesion
      flocking
    ]
  ]
end

to avoidance
```

```
  turn subtract-headings heading ([heading] of nearest-neighbour)
end

;;Keep own heading pointed towards the average heading of other boids
to cohesion
  let xdir sum [dx] of neighbours
  let ydir sum [dy] of neighbours
  ifelse (xdir = 0 and ydir = 0)
  [turn 0]
  [turn subtract-headings (atan xdir ydir) heading]
end

;; move towards average position of flockmates
to flocking
  let xdir mean [sin (towards myself + 180)] of neighbours
  let ydir mean [cos (towards myself + 180)] of neighbours
  ifelse (xdir = 0 and ydir = 0)
  [turn 0]
  [turn subtract-headings atan xdir ydir heading]
end

to turn[dir]
  ifelse (abs dir > max-turn)
  [ ifelse (dir < 0)
    [ lt max-turn]
    [ rt max-turn]
  ]
  [rt dir]

end
```

6.2 Gama code

6.2.1 Implementation 1

```
/**
* Name: boids
* Based on the internal empty template.
* Author: Leon
* Tags:
*/


model boids

/* Insert your model definition here */
```

```
global {
        int number_of_birds <- 100;
        int min_distance_from_others <- 100;
        init {
                create birds number: number_of_birds;
        }
}

species birds skills: [moving] {
        point velocity <- {0,0};
        float speed max: 1.0;
        container<birds> neighbours;
        aspect base {
                draw triangle(1) color: rgb(rnd(255), rnd(255), rnd(255)) rotate: heading;

        }

        init setup{
                write "hello" + self;
                neighbours <- birds - self;
        }

        reflex go{
                do avoidance;
                do alignment;
                do flock;
                if(((velocity.x)as int)=0) and (((velocity.y)as int)=0)
                {
                        velocity <- {rnd(1)-1, rnd(1)-1};
                }
                point prev_pos <- location;
                do goto target: location + (velocity*speed);
                velocity <- location - prev_pos;

        }

        action avoidance{
                //Make birds not crash into eachother
                point dir <- {0.0,0.0};
                ask (neighbours overlapping(circle(min_distance_from_others))){
                        dir <- dir-((location)- myself.location);
                }
                dir <-  normalize(dir);
                velocity <- velocity + dir;
```

```
        }
        action alignment {
                //Make birds rotate into the same direction as eachother
                point dir <- mean(neighbours collect(each.velocity))-velocity;
                dir <- normalize(dir);
                velocity <- velocity +dir;

        }
        action flock{
                //Make birds move towards the center of the flock
                point center <- (length(neighbours)>0)? mean (neighbours collect (each.locat
                point dir <- center-location;
                dir <- normalize(dir);
                velocity <- velocity + dir;
        }

        action normalize (point vector)
        {

                float len <- sqrt(vector.x^2+vector.y^2);

                if(len = 0)
                {
                        return vector;
                }

                point unit_vector <- {vector.x/len, vector.y/len};

                return unit_vector;
        }
}

experiment flocking {
        output{
                display boids_display{
                        species birds aspect:base;
                }
        }
}
```

6.2.2 Implementation 2

```
/**
* Name: boids
* Based on the internal empty template.
* Author: nerzu
```

```
* Tags:
*/


model boids

/* Insert your model definition here */

global torus:false {
	//Bounds
	int map_size <- 1000;
	int min_dist <- int(map_size / 20);
	int max_dist <- map_size-min_dist;
	geometry shape <- square(map_size);
	int number_of_boids <- 10 min: 1 max: 100;

	init{
		create boids number: number_of_boids {
			location <- {rnd(map_size-2)+1, rnd(map_size-2)+1};
		}
	}

}

species boids skills: [moving] {
	float speed max: 10.0;
	//Why does GAMA not have a concept of a vector? Heaven knows.
	point velocity <- {0,0};
	container<boids> neighbours;

	//I want my boids to see all other boids, so we can use this to only do it once
	init gather_neighbours{
		neighbours <- boids - self;
	}

	reflex movement{
		container<boids> nearby <- neighbours overlapping (circle(20));
		//write (string(length(neighbours)) + " " + string(length(nearby)));
		if (length(nearby) > 0)
		{
			do separate(nearby);
		}
		else {
			do alignment;
			do cohesion;
		}
```

```
        do bounding;

        if (abs(velocity.x) <= 0.5 and abs(velocity.y) <= 0.5)
        {
                velocity <- normalize({rnd(2)-1, rnd(2)-1});
        }
        point current_loc <- copy(location);
        do goto target: location + (normalize(velocity)*speed);
        velocity <- location - current_loc;
}

action separate(container<boids> others) {
        point dir <- {0,0};
        ask(others)
        {
                dir <- dir - (location - myself.location);
        }
        velocity <- velocity + normalize(dir);
}
action alignment {
        point dir <- mean(neighbours collect(each.velocity))-velocity;
        velocity <- velocity +normalize(dir);
}

action cohesion {
        point center_of_flock <- (length(neighbours)> 0) ? mean(neighbours collect
        (each.location)) : location;
        point dir <- center_of_flock - location;
        velocity <- normalize(dir);
}

action bounding {
        if  (location.x) < min_dist {
                        velocity <- velocity + {min_dist,0};
                } else if (location.x) > max_dist {
                        velocity <- velocity - {min_dist,0};
                }

                if (location.y) < min_dist {
                        velocity <- velocity + {0,min_dist};
                } else if (location.y) > max_dist {
                        velocity <- velocity - {0,min_dist};
                }
}

//Couldn't find a function to do this for me, so I have to do it myself
```

```
        point normalize(point vec) {
                float len <- sqrt(vec.x^2 + vec.y^2);
                if(len <= 1)
                {
                        return vec;
                }
                point out <- {vec.x/len,vec.y/len};
                return out;
        }

        aspect basic {
                draw triangle(20)color: #aquamarine rotate: heading;
        }
}

experiment flocking {
        float minimum_cycle_duration <- 0.01;
        output{
                display Boids_flocking type: opengl {
                        species boids aspect: basic;
                }
        }
}
```

6.3 Repast Code

6.3.1 Implementation 1

Not Available due to Workspace corruption and subsequent purge

6.3.2 Implementation 2

Not Available due to Workspace corruption and subsequent purge

6.4 Cultsim Code

6.4.1 Implementation 1

```
-- boids.lua
scenario = Scenario:new()

scenario.name = "boids 2"
scenario.description = "Boids again for testing time to implement"
scenario.bounds = Vec2:new(100.0,100.0)

scenario.systems = {
        "SensorSystem"
```

```
        systems.boid_system,
        systems.move_system,
        "DeletionSystem",
}

scenario.data_collectors = {
    data_collectors.velocity_collector
}

scenario.sampling_rate = 1.0

scenario.init = function()
        for i = 1, 20 do
                local boid = cultsim.spawn("boid")
                local boid_comp = cultsim.get_component(boid,component.lua1)
                boid_comp.velocity.x = random:uniform(-1.0,1.0)
                boid_comp.velocity.y = random:uniform(-1.0,1.0)
        end
end

scenario.update = function(dt) end
scenario.draw = function() end
scenario.shutdown = function() end

--01_data_collection.lua
data_collectors = {
    velocity_collector = DataCollector.new("Position X", {
        accum = 0.0,
        execute = function(self)
            local view = cultsim.view({component.position})
            local sum_x = 0.0

            view:each(function(e)
                local boid = cultsim.get_component(e, component.position)
                sum_x = sum_x + boid.position.x
            end)

            return sum_x / view:size()
        end
    })
}

--00_systems.lua
systems = {

        boid_system = {
```

```
        update = function(dt)
                local components = {component.position, component.lua1}
                local view = cultsim.view(components)

                local avg_pos = Vec3:new{0,0,0}
                local avg_vel = Vec2:new{0,0}

                view:each(function(e)
                        local pos_comp = cultsim.get_component(e,
                        component.position)
                        local boid_comp = cultsim.get_component(e,
                        component.lua1)

                        avg_pos.x = avg_pos.x + pos_comp.position.x
                        avg_pos.y = avg_pos.y + pos_comp.position.y

                        avg_vel.x = avg_vel.x + boid_comp.velocity.x
                        avg_vel.y = avg_vel.y + boid_comp.velocity.y
                end)

                avg_pos.x = avg_pos.x / view:size()
                avg_pos.y = avg_pos.y /view:size()

                avg_vel.x = avg_vel.x/view:size()
                avg_vel.y = avg_vel.y/view:size()

                view:each(function(e)
                        local pos_comp = cultsim.get_component(e,
                        component.position)
                        local boid_comp = cultsim.get_component(e,
                        component.lua1)

                        boid_comp.velocity.x =boid_comp.velocity.x +
                        (dt*(avg_vel.x + (avg_pos.x-pos_comp.position.x))/2)
                        boid_comp.velocity.y =boid_comp.velocity.y +
                        (dt*(avg_vel.y + (avg_pos.y-pos_comp.position.y))/2)
                end)
        end
},
move_system = {
        update = function(dt)
                local components = {component.position, component.lua1}
                local view = cultsim.view(components)

                view:each(function(e)
```

```
                                local pos_comp = cultsim.get_component(e,
                                component.position)
                                local boid_comp = cultsim.get_component(e,
                                component.lua1)

                                pos_comp.position.x = pos_comp.position.x +
                                (boid.speed * boid.velocity.x * dt)
                                pos_comp.position.y = pos_comp.position.y +
                                (boid.speed * boid.velocity.y * dt)
                        end)
                end
        }
}

-- Set system metatable for all systems
for k,v in pairs(systems) do
    setmetatable(v, {
        __index = cultlib.LuaSystem,
        __tostring = function(table) return k end
    })
end

--entities/boids.lua
entity = {
        PositionComponent = {
                position = Vec2:new(random:uniform(-scenario.bounds.x, scenario.bounds.x),
                                    random:uniform(-scenario.bounds.y, scenario.bounds.y))
        },
        SpriteComponent = {
                color = Vec3:new(random:uniform(0,255),random:uniform(0,255),
                                random:uniform(0,255)),
                texture = "sprites/arrow_c.png",
                normal = "sprites/arrow_n.png",
                scale = 10
        },
        VisionComponent = {
                radius = 50.0,
                fov = 0
        },
        TagComponent = {
                tags = ETag.Avoidable | ETag.Creature
        },
        LuaComponent = {
                lua_id = 1,
                speed = 80.0,
                velocity = Vec2:new(0,0)
```

```
        }
    }
```

B APPENDIX B; Gantt Diagram of Project plan

The following Gant diagram was created with the free template provided at google docs, found here[3]. **I did not create this template, and claim no ownership over it**. I simply used it to speed up the process of making a Gantt-diagram. All credit goes to the providers of the template, and smartsheet.

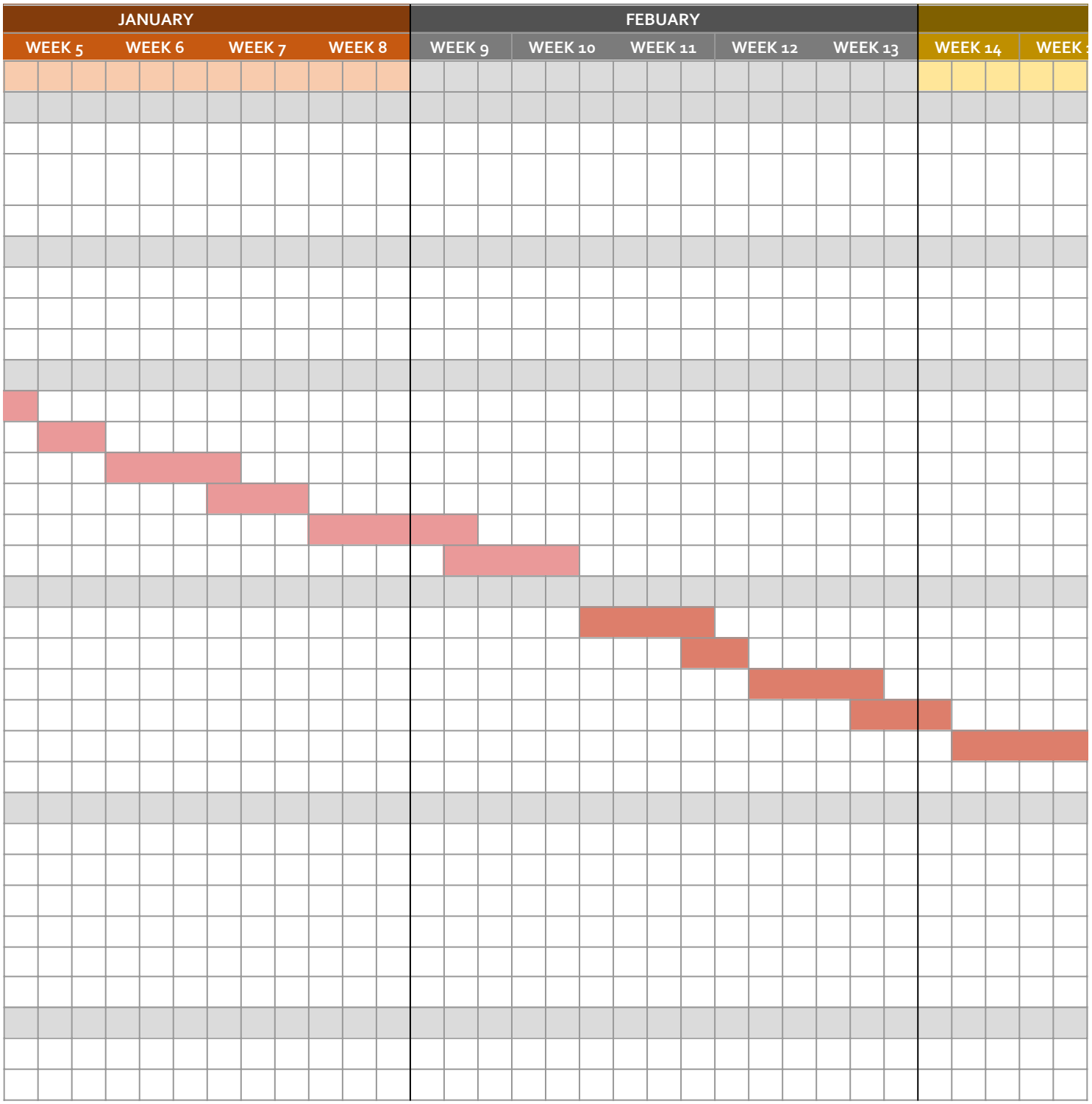
I am sorry about the formatting, but it was the best I could do while trying to keep the text readable.

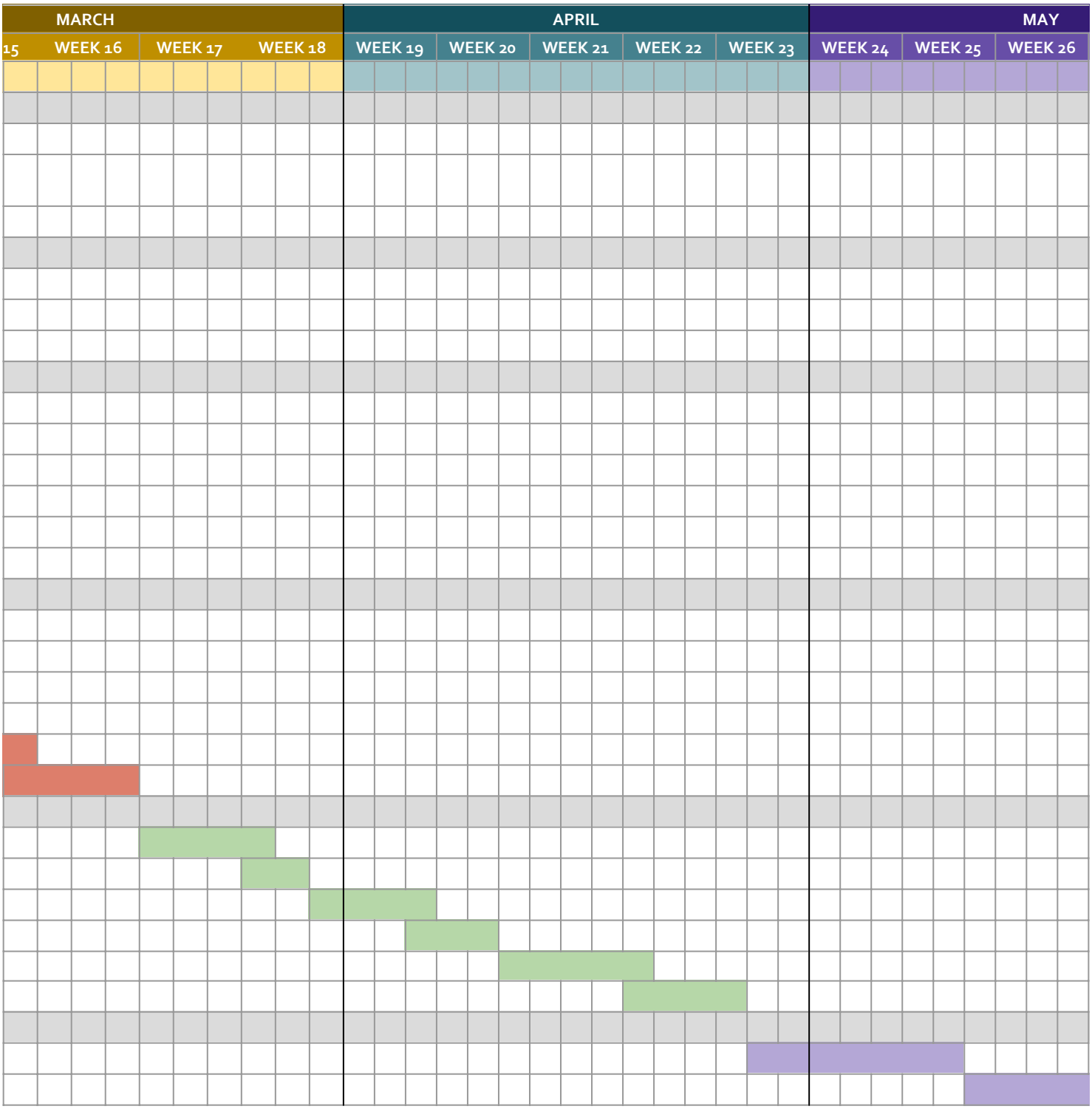
smartsheet Chart created with template supplied by smartsheet

PROJECT TITLE	Fictional History Simulator	PROJECT CREATOR	Leo
DATE	10.12.2021		

TASK REFF	TASK TITLE	DECEMBER																		
		WEEK 1					WEEK 2					WEEK 3					WEEK 4			
1	Advanced Project Work																			
1.1	Creation of Rapid prototype																			
1.2	Creation of more complete, low fidelity prototype																			
1.3	Report writing																			
2	Initial implementation																			
2.1	Further familiarization with Frameworks																			
1.5	Expanding/rewriting of initial prototype																			
1.6	Testing for validity of results of project core																			
3	Creation of Module 1																			
3.1	Researching of Relevant fields																			
3.2	Initial Rapid Prototype																			
3.3	Implementation																			
3.4	Unit tests																			
3.5	Integration																			
3.6	Integration tests																			
4	Creation of Module 2																			
4.1	Researching of Relevant fields																			
4.2	Initial Rapid Prototype																			
4.3	Implementation																			
4.4	Unit tests																			
4.5	Integration																			
4.6	Integration tests																			
5	Creation of Module 3																			
5.1	Researching of Relevant fields																			
5.2	Initial Rapid Prototype																			
5.3	Implementation																			
5.4	Unit tests																			
5.5	Integration																			
5.6	Integration tests																			
9	Writing of Master thesis																			
8.1	Inital draft																			
8.2	Finished report																			

ɔn Nicola Cinquemani




MARCH
APRIL
MAY
15
WEEK 16
WEEK 17
WEEK 18
WEEK 19
WEEK 20
WEEK 21
WEEK 22
WEEK 23
WEEK 24
WEEK 25
WEEK 26

WEEK 27			WEEK 28		

Paper II

The tool study conducted as part of the IMT4134 - Specialisation in Software Engineering course in order to find the most suitable Agent based modeling frameworks for the project.

Specialization in Software Engineering

Leon Cinquemani

December 2021

Abstract

Agent-based modeling is a sub-discipline of Simulation which focuses on modeling systems from the bottom up, as a series of interactions between agents, each of which has a series of potential actions it can take, and a series of circumstances in which it takes those actions.
Agent-based modeling is commonly used across a variety of disciplines, from flow management to social sciences, traffic-control, and organizational protocol development. However, scientists specializing in these fields might not necessarily have the programming know-how to create a model from scratch.

This is where Frameworks come into play, simplifying the process of creating models by providing a suite of tools to define agents, the actions they can take, and the metrics which need to be collected from the simulation.
Finding the right framework for the task can be challenging, as a large variety of different frameworks of varying prevalence are available, both freely available to the general public and as proprietary software.

This paper focuses on a small subset of Frameworks for Agent-Based Modeling and Simulations, evaluated by a variety of qualitative, subjective metrics, based on the author's experiences with each of the tested frameworks, and hopes to give insights into the perceived shortcomings of the different frameworks.

Contents

1 Introduction

1.1 Background

The field of Agent-based modeling is fairly wide, the concept of building a system from the ground up having found use in a variety of different fields, from social sciences to flow management.[1] As such, interest in the field has only been increasing since its inception, as more and more fields become aware of the benefits of modeling systems from the bottom up.

Agent-based modeling focuses on creating a system comprised of agents, each of which carries data and methods, and interacts with the environment it has been placed in, as well as other agents present. This allows Agent-Based Modelling and Simulation (ABMS) to represent complex interactions between a multitude of actors in a natural way, and to provide insights into phenomena that rely on the interactions of individuals, which might be difficult to accurately model mathematically.

With the applicability of agent-based modeling being very broad[1], and interest in the field expanding, it is important to provide researchers the tools they need to create their own models, adapted to the specific issues they are researching. For this reason, a large variety of ABMS Frameworks exist, which focus on making the creation of agent-based models as easy as possible, while at the same time trying to (generally) remain universally applicable for all potential use-cases of ABMS.

Since Agent-based modeling and simulation is a mature tradition of simulation by this point, there are a lot of varying frameworks that attempt to provide this assistance to intrepid researchers, of varying levels of complexity and specialization[2].
This does mean that researchers might be initially overwhelmed by the number of options available, as the right tool for the job might not immediately be clear. Studies, such as the one conducted by Abar et Al[2], seek to classify and simplify the field by sorting the available frameworks via a variety of criteria. Others, such as this paper and the study by Railsback et Al[3], focus on a small subset of these frameworks, and how they compare against one another in a variety of metrics.

1.2 Definitions

An ABMS framework, put simply, provides a set of standards and software tools[3] for developers to create their own implementations of simulation models, which can then be run and analyzed to gather insight into the underlying rules of agent interactions.
For the sake of this paper, I will be considering both frameworks such as NetL-

ogo, which come with an integrated IDE, and frameworks such as GAMA and Cultsim which come without their own IDE but can also be used with an IDE to write their models.

The test model I will be implementing in each framework is the classical, simple boid model, that is, a flocking behavior simulation in which each agent attempts to:

1. Keep a certain minimum distance from every other agent
2. Orient it's heading to be equal to the average heading of its neighboring agents
3. Maneuver in such a way as to be as close as possible to the center-point of its neighbors' positions

Each agent moves in 2D space at a constant maximum speed and can turn a specific number of degrees each turn to correct its heading based on the aforementioned criteria.

1.3 Research Questions

The reason for me choosing to undertake this project is already substantiated above, as the applicability of Agent-based models and the need to find a functional, robust framework to work with necessitate the researching of the differences between these ABM platforms to allow researchers and hobbyists to make informed decisions when choosing which framework to use when developing their models.

For my project specifically, I also had the incentive that the overarching project of my master thesis would be the implementation of a model, which meant that I found myself in a situation where I needed to choose a framework from the list of available options. As such the project also has a pragmatic reason for me. Overall the Research questions I seek to answer over the course of this project are:

1. Which Frameworks are the most user-friendly in terms of availability of online help and quality of documentation?
2. How long does it take to implement a pre-determined model for somebody that has little or no experience with the framework.

To get a broad overview of these issues, I chose a variety of Frameworks, originally examining a total of 5, with 4 of these ending up being tested, and 3 producing a full implementation of the Boids model. This was due to issues with some of the frameworks tested either providing insufficient information to successfully operate them, or being too poorly maintained to be effectively used.

2 Methodology

2.1 Finding Frameworks

For the process of finding frameworks for this project, I drew initial inspiration from a Wikipedia article on ABMS frameworks[4], where I first picked up MASON, Netlogo, Repast, and GAMA. These choices were further refined and confirmed after looking over the classifications provided in the paper by Abar et Al[2], where the previously mentioned Frameworks were considered to be capable of running models of medium to high intensity, and varied widely in how much effort went into creating a model, with Netlogo being considered the easiest to work with, and MASON and Repast both being considered difficult to model in.

These 4 frameworks were chosen based on several criteria:

- The Framework must be available to the public free of charge
- The Framework must be fully available on a Windows-based device
- Development and maintenance on the framework must still be active

I deemed these as reasonable requirements for a Framework to be used by a novice modeler, as they ensure a low barrier of entry, with the average beginner likely unwilling to spend money on something they have no experience in, and with Microsoft being used by 80+% of all personal computers.
The final criteria were added to ensure that the framework would be decently modern, following current best practices and utilizing modern technologies, to thus provide a contemporary look at the state of AMBS frameworks.
These constraints also helped choose a specific version of the Repast framework, as Repast Symphony was the only version available that satisfied all three criteria.

Additionally to these 4 frameworks, I decided to also include an in-house ABMS framework, developed by myself and a pair of fellow students, named Cultsim[5]. Cultsim was developed to allow for high-quality visualization of simulations and made to be platform-independent.
It was included both as a potential candidate for future work for myself, and as it fits most of the criteria mentioned above, except for a lack of maintenance since its initial completion.

The final 5 chosen frameworks were thus:

- Cultsim[5]
- MASON[6]
- Netlogo[7]

- Repast Symphony[8]
- GAMA[9]

2.2 Testing

To measure the frameworks, I looked at several metrics that I deemed to be significant based on the above research questions, some of which were easily quantifiable and some were not. Due to the nature of these tests, they ultimately are partially subjective, as a number of metrics are based on my personal experiences with the frameworks in question.

For this paper, I focused on the following criteria:

- Implementation
  - Time taken to implement boids when never having used the framework before
  - Time taken to re-implement boids after having done so before
- Documentation
  - How easily available online documentation for the framework is
  - The quality of the documentation based on:
    - Examples of use
    - Ease of navigation
- Assistance
  - Number of related questions on Stack-overflow
  - Number of Google search-results
  - Number of video-results on Youtube
  - Number of Papers citing the framework

2.2.1 Implementation

For the purposes of Implementation, I will be creating a model for Boids, as discussed above, from scratch for each of the frameworks. This implementation will be completed twice, and the time taken for both attempts will be recorded. Ideally, the first implementation will show how long it takes to implement a simple model with no background in the framework, and the second will give insights into how long it takes to implement the same model when the researcher knows how to do it. This should help provide an overview of how easy it is to work with the framework once one is familiar with how it functions.

There are some issues with this, however, as skills learned while working with one framework might translate over to the next, thus meaning that the

tests are not done entirely in a vacuum, as they ideally would be. As I am but one researcher with limited time to complete these tests, I have found no way of preventing this from occurring. Therefore, I will present the test results in order, and highlight any potential influences on implementation time from sources other than simply learning the framework.

2.2.2 Documentation

Documentation includes all official documentation that comes with the software, both installation instructions, step-by-step implementation guides, and reference manuals.
These will be rated based on how easy they are to find, how clear they are (how well the functionality of different parts of the framework is explained), and how easy it is to find documentation for individual primitives and functions that may be relevant to the model.

I would also report on the completeness of the documentation, as to say how much of the framework's functionality is documented, but that would require a deeper understanding of the frameworks than I possess, seeing as this study is my first interaction with most of these systems. As such, I cannot say if any parts of the system have been left out of the documentation unless it is blatantly obvious.

2.2.3 Assistance

In this category I will discuss how easy it is to find help for the framework in question from non-primary sources, that is sources outside of FAQs and documentation provided by the developers. For this study, I quantify this metric as the number of solved problems on Openstack that can be found by searching for relevant keywords, the number of google searches that come up when looking for the framework, and the number of papers published about the framework in question (if available on the developers' site).

I hope that this gives a fairly detailed overview of non-primary sources of information that can be found about the framework, and that would be intuitive for novices in the field to look at.

3 Results

3.1 Installation

To utilize these frameworks, step 1 is obviously to install the framework. This task is more involved for some of them than others, and one of the frameworks

managed to get itself removed from further testing at this step. In cases where it was required, I looked online for assistance with installing the frameworks, both from official and unofficial sources.

3.1.1 Netlogo

Netlogo was easy to install, as the installer contained everything required to run the program from the get-go. As such, installing the framework required no further assistance of any kind. The web page is a little confusing, as it contains fields to enter one's name, organization email, and any comments one may have, but these fields are entirely optional for downloading, it would seem.

The website also allows users to choose a specific version of Netlogo, but as I have no requirements for versioning, I left it on the default selected value. After downloading the installer, I only needed to choose an installation folder, and the program was installed and working.

Out of all programs surveyed, Netlogo was the easiest to get running, and the other frameworks definitely could benefit from making attempts at a similar user experience.

3.1.2 Gama-platform

Gama was roughly on the same level as Netlogo when it came to its installation, only requiring the user to choose an appropriate version. As a Java illiterate I chose the version with a built-in Java Development Kit so I did not have to worry about it, and then download the latest version of the framework.

From there, I received a .zip file that could simply be unpacked in a location of my choice and could be run from there, problem-free. Not as clean as a dedicated installer, but still very easy.

3.1.3 Repast Simphony

Repast also was simple to install via an installer, which was very appreciated. An installer is available for download from their website, which upon opening easily walks users through the steps of setting up the program, including installation destination, and whether to include a dedicated installation of the JDK, which I checked "yes" on.

The program otherwise requires little to no setup. Overall, another very smooth installation experience, slightly lessened by the fact that the installation only seems to run when booted from the start menu, for hitherto unknown

reasons. Additionally, the installation present on my machines seems to randomly decide that my work-space is corrupted, and purging the files present in it. I blame this for the lack of implementation-code available for Repast in the Appendix, and sincerely apologize.

3.1.4 MASON

Where the other publicly available frameworks were easy to install either via a simple .zip file download or via an installer, I could not get a functioning version of MASON to install itself on any machine I attempted it on. The download site presents a variety of files to download, from a .jar file which supposedly functions as a binary distribution, to a full source distribution.

According to the site, the latter of these two is the common installation, so I chose that one (though I attempted using both). This presents me with several files, including a Readme, which I next consult for what to do. It speaks of a "jar" folder, in which the file with which to run the program should be located, but the distribution does not seem to contain such a folder unless it is buried deep within the folder structure, which seems as though it would be a rather strange design choice.

The file also speaks at length of adding libraries to your Classpath, but as somebody that has little to no know-how of how Java works, it becomes clear to me that while I can follow tutorials online there is little to no way for me to see if what I did actually worked.

Likewise, the Manual for MASON, provided on their website is rather unhelpful and serves seemingly to remind me that I am not the intended audience, as it repeatedly refers me to different frameworks, namely Netlogo and Repast. I spent a good few hours trying to troubleshoot the problem to no avail, and am forced to remove MASON from the list of frameworks I am testing from this point forward, as my know-how with Java applications is seemingly not good enough to even install the program.

3.1.5 Cultsim

Cultsim is a similarly sad tale to MASON when it comes to user convenience when downloading and installing the file. The entire project is available via a GitHub link[5] and can either be cloned as a source-code project or be downloaded as a set of binaries. I tried both and found the latter to be far easier to get working, as the source distribution struggles from poor maintenance and several dependencies have since been updated to include errors that break the project.

Via the binaries, I receive a simple .zip file, and after unpacking am greeted with a ".exe" file, as well as a few others, including a data.zip file. Unzipping that as well makes the program functional, and it runs just as the source distribution does, albeit with less of a headache to get it up and running.

Overall, the binary installation was entirely pain-free up until this point. The installation instructions on Github were also concise and informed me of what to do to get the program running step by step. I would still say it is the most convoluted to get set up before running out of the four frameworks that I managed to install, but by comparison to MASON, its installation is relatively painless.

Installation alone has already removed one of the 5 frameworks from further testing, which means that from this point forwards, MASON will no longer be considered for the rest of the study. Still, I found it valuable to include an example of a poorly realized, convoluted, and confusing installation process, as I could not get the program to correctly install, even by searching for online assistance for installation. This, coupled with the less than helpful user manual and Read me make me feel justified in dropping the framework at this point, as I doubt that somebody new to ABMS frameworks would continue troubleshooting, and rather swap to another option.

3.2 Boids

From this point, I continued with the 4 remaining frameworks and proceeded to attempt to implement the Boids framework in each of them. While the exact implementation varied (Complete code for each implementation of Boids available in the appendix below), the pseudocode could be described as follows:

```
//step 1: Setup
Reset world
Create x agents with x = random & y = random

//step 2: Per tick
For every agent:
    update_direction()
    move()

//update_direction
neighbours = other agents within radius
if(distance to closest agent < minimum distance)
separate()
else
alignment()
cohesion()
```

```
//separate
direction = direction towards closest neighbour + 180
return direction

//alignment
heading = heading of each neighbour
heading = heading / neighbours.size
direction = towards heading
return direction

//cohesion
x = average x coordinate of neighbours
y = average y coordinate of neighbours
direction = towards [x,y]
return direction

//move
set heading = direction
move_forwards(speed)
```

The implementation varies from Framework to framework, but overall stays true to this formula.
As mentioned previously this test was implemented twice in each framework, in the hopes of capturing both time required to learn the framework, as well as how long it would take to implement once one is familiar with the said framework. To expose any potential biases towards having an easier time learning later frameworks due to experience gained from earlier frameworks, these results will be presented in order of implementation, and the results will be discussed below.

3.2.1 Cultsim

As can be noted in figure 1, Cultsim does not have any data associated with it. This is because I found myself unable to implement Boids as per the above specifications into Cultsim as is, as I would have to modify the source code of the framework to allow for finding the distance and direction to the closest neighbor, and thus am unable to implement separation.

The example implementation of Boids within Cultsim also does not include separation, and thus I am unable to fully implement Boids to the specifications set. Additionally, even when I attempted to implement the less complete version of Boids without separation, my code refused to run, even when I created a near 1-1 copy of the Boids code supplied with the framework. Neither the version created for the binaries nor the version in the source distribution managed to run without crashing.

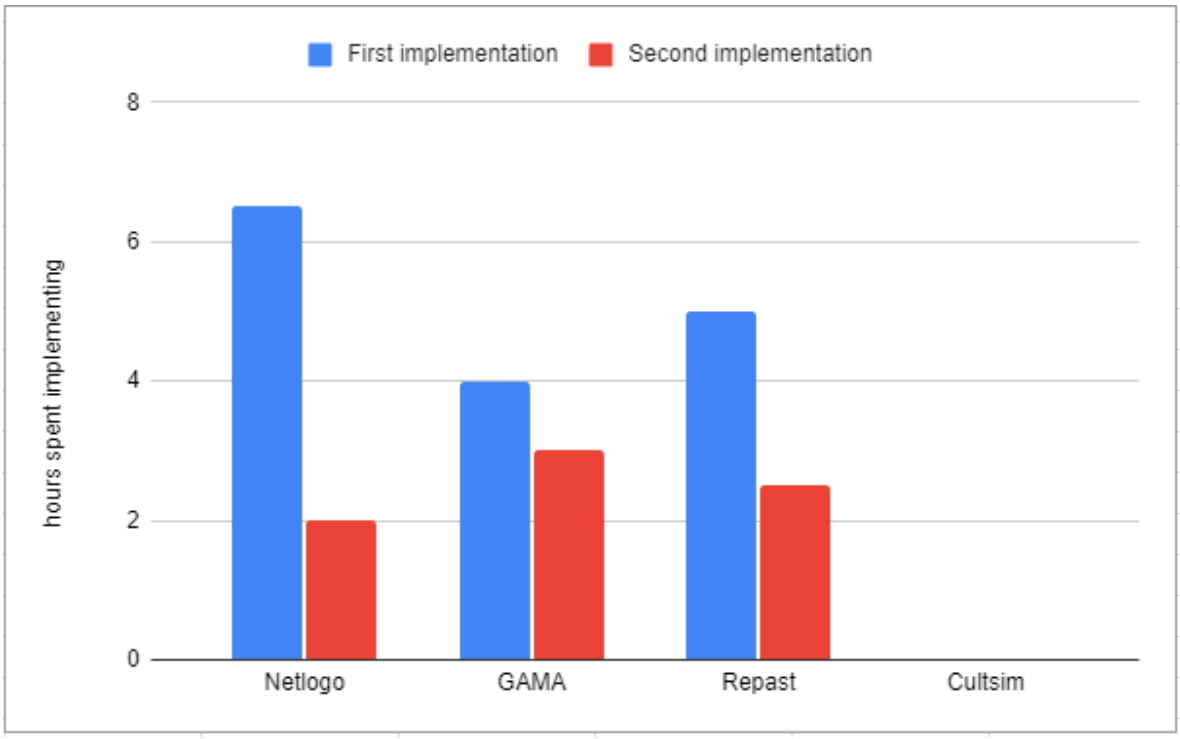


Figure 1: Time to implement Boids in Hours

I found no easy fix for the issue in the (unavailable) documentation and therefore was forced to abandon Cultsim for this test at this point. The binary version did not seem to allow for users to create their own scenarios or required additional steps not mentioned in the documentation and unknown to even me, one of its creators.
The source code version did allow me to implement my own scenarios after I resolved most of the dependency errors, but continued to crash, even though the source code was near identical to the code of the example implementation.

I, therefore, deemed it unlikely that somebody with no further knowledge on AMBS frameworks or Cultsim would attempt to troubleshoot much further, especially given the seemingly broken documentation of Cultsim, which does not build when following instructions from the source distribution, and is not included in the binary distribution, which seems like another oversight.

3.2.2 Netlogo

Netlogo was touted in every source I came across as the most beginner-friendly of the frameworks, and thus I decided it would be a good starting point for the project. The first attempt at implementing Boids cost me roughly 4 hours and did not end up functioning as intended. This was due to several logical errors made in the code of the model itself, which ended up taking a lot of time to fix.

Luckily, it was at this point that I noticed the example projects that come included with Netlogo, the likeness of which are also included in each of the other frameworks that I tested. This made the implementation of the code a lot easier overall as I now had an example to look at for how to do the more complex parts of the Boids implementation and the syntax for the framework. This drastically sped up working times, and I managed to finish the implementation of the model within an additional 2.5 hours.

This time was drastically reduced for the second attempt, which ended up merely taking 2 hours from start to finish, the shortest time for re-implementation noted during the tests, producing an equivalent implementation in less than 1/3 of the time it took to originally create the Model.

3.2.3 GAMA

The Gama implementation of the Project proved to be relatively painful to complete, mostly due to the framework's documentation lacking several key concepts that I would have expected from a framework of this sort, such as the concept of a vector, vector normalization, and similar concepts. The example implementation of Boids also proved to be rather crude, as it did, in fact, **not** normalize its vectors, instead using several multipliers to allow users to set how heavily separation, alignment, and cohesion are weighted.

While one might say that this allows users to steer the importance of the individual components of the flocking behavior, I would argue that the same level of interactivity could have been achieved better by first normalizing the vectors, and then multiplying them with a multiplier between 0 and 1.
That at least is my opinion on the matter, others may disagree. As user interactivity was not a part of the pseudo-code requirements, I did not bother to implement it.

The second time around still took an inordinate amount of time, as the way Gama handles its model implementations is rather cryptic and the documentation is, again, rather poor. Overall, the time taken only went down from 4 hours for the initial implementation to 3 hours for the second attempt. This makes GAMA the slowest framework when it comes to re-implementation.

3.2.4 Repast Simphony

Repast is a very complex program to set up, and documentation for getting the program up and running is rather sparse, with the tutorials being a little more cryptic than what would be appreciated. Online video tutorials for Repast were largely nonexistent, and the written setup tutorial had screenshots that did not apply to my installation of the program, making finding the correct buttons for setting up the project somewhat difficult, especially as a ReLogo Project and a Repast Project look very similar if one is not looking carefully.

With the initial hurdles overcome, however, Repast proves itself to be a rather straightforwards framework, aided by the fact that ReLogo, the programming language used by Repast, is closely related to the language employed by Netlogo, and thus a lot of syntax and functionality carried over. This made it easy to implement the test once I understood that coding for Repast, in the simplest sense, was a lot like coding for Netlogo in java.

This drastically sped up the process of implementation, and the second test only took a little longer than implementing in Netlogo, despite the additional overhead from having to implement visualization, which Netlogo does automatically.

3.3 Documentation

3.3.1 Netlogo

Netlogo's documentation is easily the best of the four, containing a complete dictionary of all concepts used by the frameworks programming language, including several tutorials, and easily being available in a variety of formats from the front page under the user-manuals section.

Netlogo's documentation provides step-by-step tutorials for implementing some simple models, guides for all key parts of the creation of a model in Netlogo, and a dictionary of all concepts, primitives, and functions available to creators in Netlogo, complete with examples of usage for the non-self-explanatory concepts.

Netlogo's documentation is as good as I have seen documentation get for any software I have worked with and stands head and shoulders over the documentation provided by its peers. For the entirely programming illiterate, I can only assume that this alone would set Netlogo far and above the rest, as it seems to me that one could create models for Netlogo simply by looking over the documentation and using the dictionary provided.

3.3.2 GAMA

While extensive, I find myself unable to praise GAMA in the same way that I praised Netlogo. There is nothing particularly wrong with the documentation, as it provides much of the same information that Netlogo provides, yet I feel that it is presented in a more cluttered, less transparent way, perhaps because each section must be opened into a drop-down menu before it's contents are shown. The way operators are handled is also less than appealing in GAMA, with them being sorted alphabetically, rather than being linked to the individual data structures they operate on.

The way that the complete dictionary is presented in GAMA also is nowhere near as intuitive as Netlogo, and I frequently found myself having to ctrl+f my way through it when attempting to find information on a specific function and variable.
The documentation feels cluttered and is harder to navigate than that of Netlogo, but it is very much serviceable overall, even if it gives me the feeling that I am missing key parts of it because I do not know how to find them.

Most concepts are explained well enough, with a majority having a small example of usage next to them, barring a few exceptions. Overall, the documentation is not as good as that of Netlogo, but still decent enough.

3.3.3 Repast Simphony

Repast's documentation is neatly divided into a lot of different files, some more easily noticeable than others. The complete reference guide seems rather off-puttingly designed again and does not include a reference for functions and similar primitives provided by Repast, but it does come with detailed explanations and screenshots for most of, if not all the key concepts of Repast.

The actual list of primitives and associated functions can be found in the Relogo primitives file, which has most of what I would want from a list of available functions and primitives, but suffers due to the fact it scrolls both vertically *and horizontally*, making the page hard to navigate.

Other than these gripes, the tutorials are well implemented and decently well-annotated, giving users a sense of direction as to where to start when learning the program, and how to progress forward through learning the framework. Overall it is serviceable and less cluttered than the Gama documentation.

3.3.4 Cultsim

Despite the assurances on the main page that Cultsim allows users to build the documentation after downloading the source code version, I have been unable

to get this functionality to work properly, creating no useful documents whatsoever.

What is worse, the binary implementation does not come with any form of documentation, and no help is available online.
Ironically, even if I managed to get the documentation to build, that would not help me in implementing any models, as the documentation for the Lua scripting, the part of Cultsim that is used for model definition is unfinished, and thus unavailable.
Meaning, that even if I had managed to build the documentation, that would only help me with changing the source code, not actually writing a model.

Thus, I once more have to give Cultsim a failing grade in this study, as the program does not provide any valuable documentation to the user beyond the example implementations present in the files, making it archaic and non-user-friendly, even if it were to function as intended.

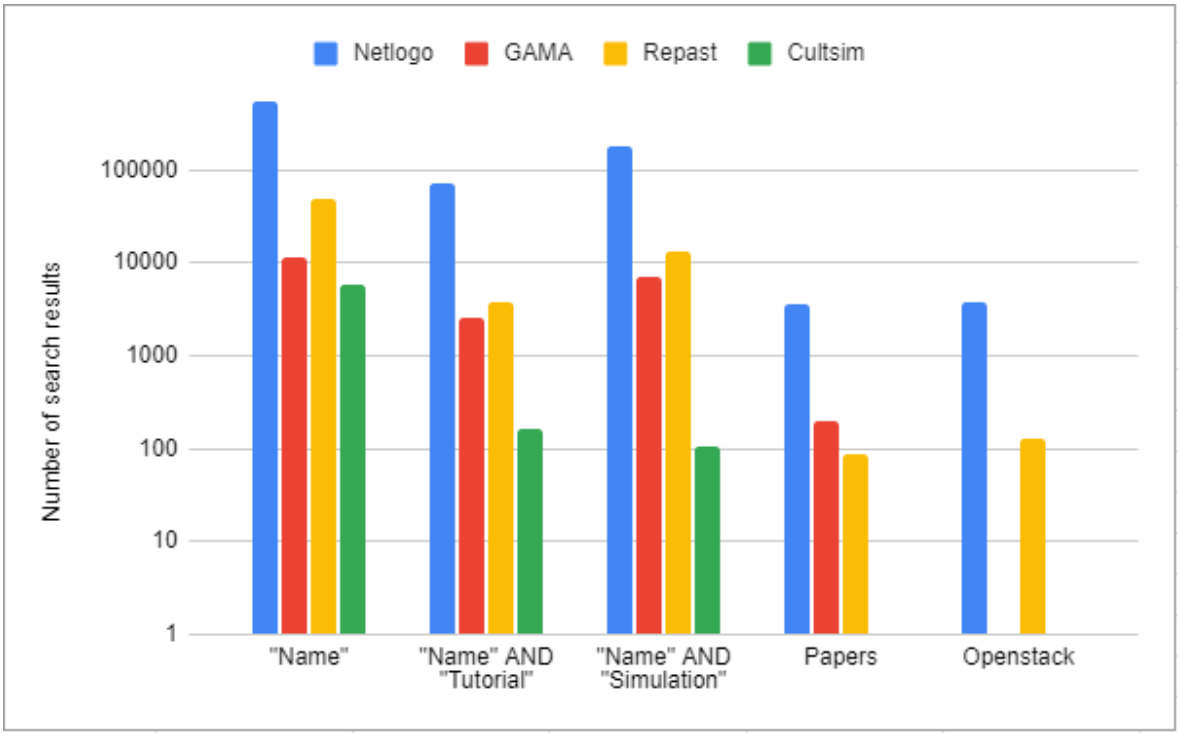


Figure 2: Search results across various sites

3.4 Assistance

In this section I will be elaborating on the amount of online help that could be found for the individual frameworks via the metrics previously defined as presented by the above figure 2, these being:

- Number of Google results
  These are separated into three different categories by the search terms used to find the results. These are "Name of Framework", "Name of Framework" AND "Tutorial", and "Name of Framework" AND "Simulation", as these seem like the most likely search queries one would use to find information and help for these frameworks specifically. The requirement of including the exact phrases requested should lower the number of false positives, that is irrelevant results being counted.
  This can however not be eliminated, as shown by the fact that Cultsim has several thousand search results, despite only having one relevant result, that being the actual GitHub page from which one can download the framework.

  Still, as I am including the irrelevant results for the other frameworks, it would feel unfair not to do the same for Cultsim. In this capacity, it also serves as a rough estimate on how many false results might be present in the searches for the other frameworks.

- Number of academic papers
  This is the number of academic papers associated with the frameworks, as denoted by the papers listed on the websites of the frameworks. This is by no means an exclusive list but should give some idea of which frameworks are most cited in academic works.
  It should also be noted that technically Cultsim does have 1 paper written about it, that being the bachelor thesis that it was created for, but that paper is not available on the Github-page that serves as Cultsim's homepage, and thus is not included, leaving its number of academical papers as 0.

- Number of questions on Stack overflow This includes both open and resolved questions, as found via the tag [Framework name]. The number is reported by StackOverflow's search results. Surprisingly StackOverflow has no active tag for the Gama platform, and no relevant results come up when searching for it in any way I have found. Thus, it shares last place with Cultsim, having no questions of Stack-overflow.

4 Discussion

4.1 Issues

4.1.1 Subjectivity of tests

Due to the less than objective nature of some of the tests, the reproducibility of my results, particularly in the fields of time required to learn the tests and the ratings of the documentation, is likely to not be very high.
Ideally, these tests would have included results from more test subjects, to get a more generalized overview of the metrics mentioned. However, I did not find myself in a position to include more testers due to concerns regarding both privacy, as well as public health and safety due to the Covid-19 pandemic currently ongoing, and the resulting lack of easy access to test subjects.

Still, I believe that the overall trends noted, and general commentary on the individual framework in every category tested has merit, and might aid in building an understanding of how the 5 frameworks discussed stack up to one another, both in terms of how usable they are, and how easy they are to pick up initially.

While by no means a definitive answer, I believe that the data provided still presents the experiences that I had when attempting to learn these 5 frameworks faithfully, and hopefully can be of assistance in choosing among these frameworks.

4.1.2 Inaccuracy of results

Due to a variety of factors, most of which I already touched on, several of the test results are likely somewhat skewed. The outside variables noted will be presented here once more in more clarity to allow for a more critical overview of what the numbers presented mean.

- Increased familiarity with the field and frameworks
  As mentioned previously, it is likely, that my methodology for researching the frameworks and implementing my test model improved over the course of the study, thus skewing the results in time taken to implement the tests in favor of the frameworks that came last.

  I think this is most clear in the time taken to originally implement Boids for Netlogo, which was before I found the example models included with each of the frameworks, which were heavily used in the following implementation work. This, in addition to me growing more familiar with how to implement Boids over time likely inflated the time taken to originally

implement the test for Netlogo significantly.

If the first, failed attempt at implementing Boids for Netlogo had not happened, I would think that the times for implementing Boids in Netlogo the first and second time would have been near-identical, as the implementing of simple models is very easy in Netlogo, as the time to re-implement the model the second time prooves.

I also think it should be noted that Repast's model was made significantly easier to implement once I realized that a lot of Netlogo's excellent documentation near-perfectly translated for ReLogo, and thus it is likely that I was able to finish the initial implementation faster because of this, and might have taken longer if I had gone through the Frameworks in the opposite order.

- False Positives for Cultsim Searches
  Unfortunately, there exists a videogame by the name of "Cultist simulator"[10], often shortened to Cultsim by its fanbase when discussing it on the internet. This means that all three search results for "Cultsim", "Cultsim" AND "Tutorial", and "Cultsim" AND "Simulation" on google are heavily inflated with results about this game, tutorials for playing the game, and the fact it is a simulator.

  There is not much that can be done to eliminate these results from the list and keep within the boundaries of the predefined search keywords. The best I have found is the adding of the "-game" keyword to the google search, slashing the results down to a third of their original number. However, including the same keyword for "Netlogo", that is "Netlogo" -games somehow returns more results than just "Netlogo".

  As such, I have decided to leave the values of Cultsim as is, and clearly stated in the paragraphs pertaining to Cultsim's search results that they are heavily inflated, in hopes of not only providing an overview of how inaccurate google results for the other frameworks might be but also to hopefully not sabotage the results of those searches with the strange quirks of the google search bar.

  I will state it again here, to make certain it is noted. The accurate number of websites that contain information on the framework Cultsim is less than 100, and likely less than 10. The heavily inflated results are included in the list merely as a measuring stick for potential inaccuracy in the other search results and as a warning against trusting google search results to be an accurate metric.

4.2 Future work

As for future work in the field, I have a few suggestions on how to continue on the work presented in this paper, as well as a few ideas that were originally meant to be part of the work done for the here presented study, but that ended up having to be cut due to time constraints.

4.2.1 More Frameworks

The first and most obvious continuation of the work here presented would be the application of the same or similar tests to a wider range of Frameworks. Ideally, at least to the point where one could get Mason to work.
Having more frameworks tested and presented would result in a better overview of the field, and thus serve as a more valuable resource to future researchers. Additionally, it would help to further highlight trends in the world of ABMS frameworks and their components, such as the ease of installation, quality of documentation, and the impact it has on how easily adopted the individual frameworks are. Seeing as Netlogo had good scores across the board, ignoring the inflated time it took to get the first implementation done, I wonder if that is specific to Netlogo, or if other characteristics can explain the outstanding performance of Netlogo, such as its integrated IDE, or it's automatic visualization, for example.

4.2.2 More Tests

Making more models to test implementation times for the frameworks would result in a richer, more conclusive set of results, and help to eliminate some of the biases from becoming accustomed to the possible issues that need to be faced when implementing the model across different frameworks.
Perhaps changing up the order in which implementations are performed for each new model would help in further alleviating these biases.

Originally I had planned to supplement the "Boids" model with models for "Food scavenging" and "Predator avoidance". However, both of these needed to be cut due to time constraints. As fairly simple problems, they seem as though they have example models in each of the frameworks tested (though those in Cultsim might not be entirely functional), and as such should make for good candidates for future tests.

4.2.3 More Testers

To reduce the subjectivity of the tests that deal with the time taken to implement the test models, as well as the metrics for how good a framework's documentation actually is, I find it likely that increasing the number of people

running these tests would be beneficial.
This does however require both the recruiting of more testers that are willing and capable of implementing these models, as well as the consideration of privacy and health concerns, all of which were far beyond the scope of this study.

5 Conclusion

Over the course of this paper, I have presented the results of roughly 3 months of studies on a variety of agent-based modeling and simulation frameworks, and implementation of a simple Boids model for each of them to study how long it takes to learn the framework, as well as how long it takes to implement the model a second time once one is familiar with the framework.

I have gone into detail on my findings on the documentation, tutorials, and outside assistance provided for each of the frameworks presented, excluding MASON, which had to be removed from further testing early due to me being unable to successfully install it on my machine by following installation instructions to the best of my ability.

I have presented my results for each of these tests, as well as discussed what could have been done better during the project, and what could be done in the future to expand on the work that I have presented.

Overall, the results point towards Netlogo and Repast being the two frameworks that are the most user-friendly, being easy to install, having good or serviceable documentation, and not taking long to model with once one has learned how they work. Additionally, I will highlight Netlogo's quick workflow, which allows for the framework to be used for rapid prototyping, which is another point in its favor. It is up to the task at hand whether the expanded toolset of Repast is worth the additional effort in implementation compared to Netlogo.

MASON and Cultsim are the least user-friendly of the frameworks, the former being difficult to install, to the point that it had to be excluded from the remainder of the study, and the latter being poorly maintained, to the point that several dependencies required for the program to run are broken.
This necessitates a lot of work from users to even get the framework up and running for both of them, and thus is not advisable for beginners or non-programmers.

I will however highlight here that out of all the Frameworks, Cultsims visual representations were of the highest quality, which was the highlighted feature of the framework. This does however not counterbalance the glaring flaws highlighted throughout the study, but I still found it worth noting.

References

[1] Eric Bonabeau. "Agent-based modeling: Methods and techniques for simulating human systems". In: *Proceedings of the National Academy of Sciences* 99.suppl 3 (2002), pp. 7280–7287. ISSN: 0027-8424. DOI: 10.1073/pnas.082080899. eprint: https://www.pnas.org/content/99/suppl_3/7280.full.pdf. URL: https://www.pnas.org/content/99/suppl_3/7280.

[2] Sameera Abar et al. "Agent Based Modelling and Simulation tools: A review of the state-of-art software". In: *Computer Science Review* 24 (2017), pp. 13–33. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2017.03.001. URL: https://www.sciencedirect.com/science/article/pii/S1574013716301198.

[3] Steven F. Railsback, Steven L. Lytinen, and Stephen K. Jackson. "Agent-based Simulation Platforms: Review and Development Recommendations". In: *SIMULATION* 82.9 (2006), pp. 609–623. DOI: 10.1177/0037549706073695. eprint: https://doi.org/10.1177/0037549706073695. URL: https://doi.org/10.1177/0037549706073695.

[4] https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software. Accessed 08.12.2021.

[5] https://github.com/Togtja/CultSim. Accessed 08.12.2021.

[6] https://cs.gmu.edu/~eclab/projects/mason/. Accessed 08.12.2021.

[7] https://ccl.northwestern.edu/netlogo/. Accessed 08.12.2021.

[8] https://repast.github.io/. Accessed 08.12.2021.

[9] https://gama-platform.org/. Accessed 08.12.2021.

[10] https://weatherfactory.biz/cultist-simulator/. Accessed 09.12.2021.

6 Appendix

6.1 Netlogo code

6.1.1 Implementation 1

```
to setup
 ca
  reset-ticks
  create-turtles num-turtles[
    setxy random-xcor random-ycor
  ]
end

to delete
  ca
```

```
  reset-ticks
end

;;Get The heading of the turtle towards the average heading of n neighbours
to align [neighbours]
  let xdir sum [dx] of neighbours
  let ydir sum [dy] of neighbours
  if xdir != 0 and ydir != 0
  [turn subtract-headings (atan xdir ydir) heading]
end

;; Get the direction towards the average position of n neighbours
to cohesion[neighbours]
  let xdir mean [sin (towards myself + 180 )] of neighbours
  let ydir mean [cos (towards myself + 180)] of neighbours
  if xdir != 0 and ydir != 0
  [turn subtract-headings (atan xdir ydir) heading]

end

to avoidance
  let neighbour min-one-of other turtles [distance myself]
  turn subtract-headings heading ([heading] of neighbour)
end


to boids[neighbours]
  ifelse (distance (min-one-of other turtles [distance myself])) < min-dist
  [avoidance]
  [align neighbours
    cohesion neighbours]
end

to turn [dir]
  ifelse dir > max-turn
  [ifelse dir > 0
    [ rt max-turn ]
    [ lt max-turn]
  ]
  [ rt dir]
end

to go
  tick
  ask turtles[
    let closest-turtles min-n-of num-neighbours (other turtles) [distance myself]
```

```
   boids closest-turtles
   forward 1
  ]
end
```

6.1.2 Implementation 2

```
turtles-own [ neighbours nearest-neighbour]


to setup
ca
  reset-ticks
  create-turtles num-turtles[
    set xcor random-xcor
    set ycor random-ycor
  ]
end

to go
  tick
  ask turtles[
  boids-heading
  forward 1
  ]
end

to clear
ca
reset-ticks
end

to boids-heading
  set neighbours other turtles in-radius radius
  if(count neighbours > 0)
  [
    set nearest-neighbour min-one-of neighbours [distance myself]
    ifelse(distance nearest-neighbour < min-dist)
    [avoidance]
    [cohesion
      flocking
    ]
  ]
end

to avoidance
```

```
  turn subtract-headings heading ([heading] of nearest-neighbour)
end

;;Keep own heading pointed towards the average heading of other boids
to cohesion
  let xdir sum [dx] of neighbours
  let ydir sum [dy] of neighbours
  ifelse (xdir = 0 and ydir = 0)
  [turn 0]
  [turn subtract-headings (atan xdir ydir) heading]
end

;; move towards average position of flockmates
to flocking
  let xdir mean [sin (towards myself + 180)] of neighbours
  let ydir mean [cos (towards myself + 180)] of neighbours
  ifelse (xdir = 0 and ydir = 0)
  [turn 0]
  [turn subtract-headings atan xdir ydir heading]
end

to turn[dir]
  ifelse (abs dir > max-turn)
  [ ifelse (dir < 0)
    [ lt max-turn]
    [ rt max-turn]
  ]
  [rt dir]

end
```

6.2 Gama code

6.2.1 Implementation 1

```
/**
* Name: boids
* Based on the internal empty template.
* Author: Leon
* Tags:
*/


model boids

/* Insert your model definition here */
```

```
global {
        int number_of_birds <- 100;
        int min_distance_from_others <- 100;
        init {
                create birds number: number_of_birds;
        }
}

species birds skills: [moving] {
        point velocity <- {0,0};
        float speed max: 1.0;
        container<birds> neighbours;
        aspect base {
                draw triangle(1) color: rgb(rnd(255), rnd(255), rnd(255)) rotate: heading;

        }

        init setup{
                write "hello" + self;
                neighbours <- birds - self;
        }

        reflex go{
                do avoidance;
                do alignment;
                do flock;
                if(((velocity.x)as int)=0) and (((velocity.y)as int)=0)
                {
                        velocity <- {rnd(1)-1, rnd(1)-1};
                }
                point prev_pos <- location;
                do goto target: location + (velocity*speed);
                velocity <- location - prev_pos;

        }

        action avoidance{
                //Make birds not crash into eachother
                point dir <- {0.0,0.0};
                ask (neighbours overlapping(circle(min_distance_from_others))){
                        dir <- dir-((location)- myself.location);
                }
                dir <-  normalize(dir);
                velocity <- velocity + dir;
```

```
        }
        action alignment {
                //Make birds rotate into the same direction as eachother
                point dir <- mean(neighbours collect(each.velocity))-velocity;
                dir <- normalize(dir);
                velocity <- velocity +dir;

        }
        action flock{
                //Make birds move towards the center of the flock
                point center <- (length(neighbours)>0)? mean (neighbours collect (each.locat
                point dir <- center-location;
                dir <- normalize(dir);
                velocity <- velocity + dir;
        }

        action normalize (point vector)
        {

                float len <- sqrt(vector.x^2+vector.y^2);

                if(len = 0)
                {
                        return vector;
                }

                point unit_vector <- {vector.x/len, vector.y/len};

                return unit_vector;
        }
}

experiment flocking {
        output{
                display boids_display{
                        species birds aspect:base;
                }
        }
}
```

6.2.2 Implementation 2

```
/**
* Name: boids
* Based on the internal empty template.
* Author: nerzu
```

```
* Tags:
*/


model boids

/* Insert your model definition here */

global torus:false {
	//Bounds
	int map_size <- 1000;
	int min_dist <- int(map_size / 20);
	int max_dist <- map_size-min_dist;
	geometry shape <- square(map_size);
	int number_of_boids <- 10 min: 1 max: 100;

	init{
		create boids number: number_of_boids {
			location <- {rnd(map_size-2)+1, rnd(map_size-2)+1};
		}
	}

}

species boids skills: [moving] {
	float speed max: 10.0;
	//Why does GAMA not have a concept of a vector? Heaven knows.
	point velocity <- {0,0};
	container<boids> neighbours;

	//I want my boids to see all other boids, so we can use this to only do it once
	init gather_neighbours{
		neighbours <- boids - self;
	}

	reflex movement{
		container<boids> nearby <- neighbours overlapping (circle(20));
		//write (string(length(neighbours)) + " " + string(length(nearby)));
		if (length(nearby) > 0)
		{
			do separate(nearby);
		}
		else {
			do alignment;
			do cohesion;
		}
```

```
        do bounding;

        if (abs(velocity.x) <= 0.5 and abs(velocity.y) <= 0.5)
        {
                velocity <- normalize({rnd(2)-1, rnd(2)-1});
        }
        point current_loc <- copy(location);
        do goto target: location + (normalize(velocity)*speed);
        velocity <- location - current_loc;
}

action separate(container<boids> others) {
        point dir <- {0,0};
        ask(others)
        {
                dir <- dir - (location - myself.location);
        }
        velocity <- velocity + normalize(dir);
}
action alignment {
        point dir <- mean(neighbours collect(each.velocity))-velocity;
        velocity <- velocity +normalize(dir);
}

action cohesion {
        point center_of_flock <- (length(neighbours)> 0) ? mean(neighbours collect
        (each.location)) : location;
        point dir <- center_of_flock - location;
        velocity <- normalize(dir);
}

action bounding {
        if  (location.x) < min_dist {
                        velocity <- velocity + {min_dist,0};
                } else if (location.x) > max_dist {
                        velocity <- velocity - {min_dist,0};
                }

                if (location.y) < min_dist {
                        velocity <- velocity + {0,min_dist};
                } else if (location.y) > max_dist {
                        velocity <- velocity - {0,min_dist};
                }
}

//Couldn't find a function to do this for me, so I have to do it myself
```

```
        point normalize(point vec) {
                float len <- sqrt(vec.x^2 + vec.y^2);
                if(len <= 1)
                {
                        return vec;
                }
                point out <- {vec.x/len,vec.y/len};
                return out;
        }

        aspect basic {
                draw triangle(20)color: #aquamarine rotate: heading;
        }
}

experiment flocking {
        float minimum_cycle_duration <- 0.01;
        output{
                display Boids_flocking type: opengl {
                        species boids aspect: basic;
                }
        }
}
```

6.3 Repast Code

6.3.1 Implementation 1

Not Available due to Workspace corruption and subsequent purge

6.3.2 Implementation 2

Not Available due to Workspace corruption and subsequent purge

6.4 Cultsim Code

6.4.1 Implementation 1

```
-- boids.lua
scenario = Scenario:new()

scenario.name = "boids 2"
scenario.description = "Boids again for testing time to implement"
scenario.bounds = Vec2:new(100.0,100.0)

scenario.systems = {
        "SensorSystem"
```

```
        systems.boid_system,
        systems.move_system,
        "DeletionSystem",
}

scenario.data_collectors = {
    data_collectors.velocity_collector
}

scenario.sampling_rate = 1.0

scenario.init = function()
        for i = 1, 20 do
                local boid = cultsim.spawn("boid")
                local boid_comp = cultsim.get_component(boid,component.lua1)
                boid_comp.velocity.x = random:uniform(-1.0,1.0)
                boid_comp.velocity.y = random:uniform(-1.0,1.0)
        end
end

scenario.update = function(dt) end
scenario.draw = function() end
scenario.shutdown = function() end

--01_data_collection.lua
data_collectors = {
    velocity_collector = DataCollector.new("Position X", {
        accum = 0.0,
        execute = function(self)
            local view = cultsim.view({component.position})
            local sum_x = 0.0

            view:each(function(e)
                local boid = cultsim.get_component(e, component.position)
                sum_x = sum_x + boid.position.x
            end)

            return sum_x / view:size()
        end
    })
}

--00_systems.lua
systems = {

        boid_system = {
```

```
        update = function(dt)
                local components = {component.position, component.lua1}
                local view = cultsim.view(components)

                local avg_pos = Vec3:new{0,0,0}
                local avg_vel = Vec2:new{0,0}

                view:each(function(e)
                        local pos_comp = cultsim.get_component(e,
                        component.position)
                        local boid_comp = cultsim.get_component(e,
                        component.lua1)

                        avg_pos.x = avg_pos.x + pos_comp.position.x
                        avg_pos.y = avg_pos.y + pos_comp.position.y

                        avg_vel.x = avg_vel.x + boid_comp.velocity.x
                        avg_vel.y = avg_vel.y + boid_comp.velocity.y
                end)

                avg_pos.x = avg_pos.x / view:size()
                avg_pos.y = avg_pos.y /view:size()

                avg_vel.x = avg_vel.x/view:size()
                avg_vel.y = avg_vel.y/view:size()

                view:each(function(e)
                        local pos_comp = cultsim.get_component(e,
                        component.position)
                        local boid_comp = cultsim.get_component(e,
                        component.lua1)

                        boid_comp.velocity.x =boid_comp.velocity.x +
                        (dt*(avg_vel.x + (avg_pos.x-pos_comp.position.x))/2)
                        boid_comp.velocity.y =boid_comp.velocity.y +
                        (dt*(avg_vel.y + (avg_pos.y-pos_comp.position.y))/2)
                end)
        end
},
move_system = {
        update = function(dt)
                local components = {component.position, component.lua1}
                local view = cultsim.view(components)

                view:each(function(e)
```

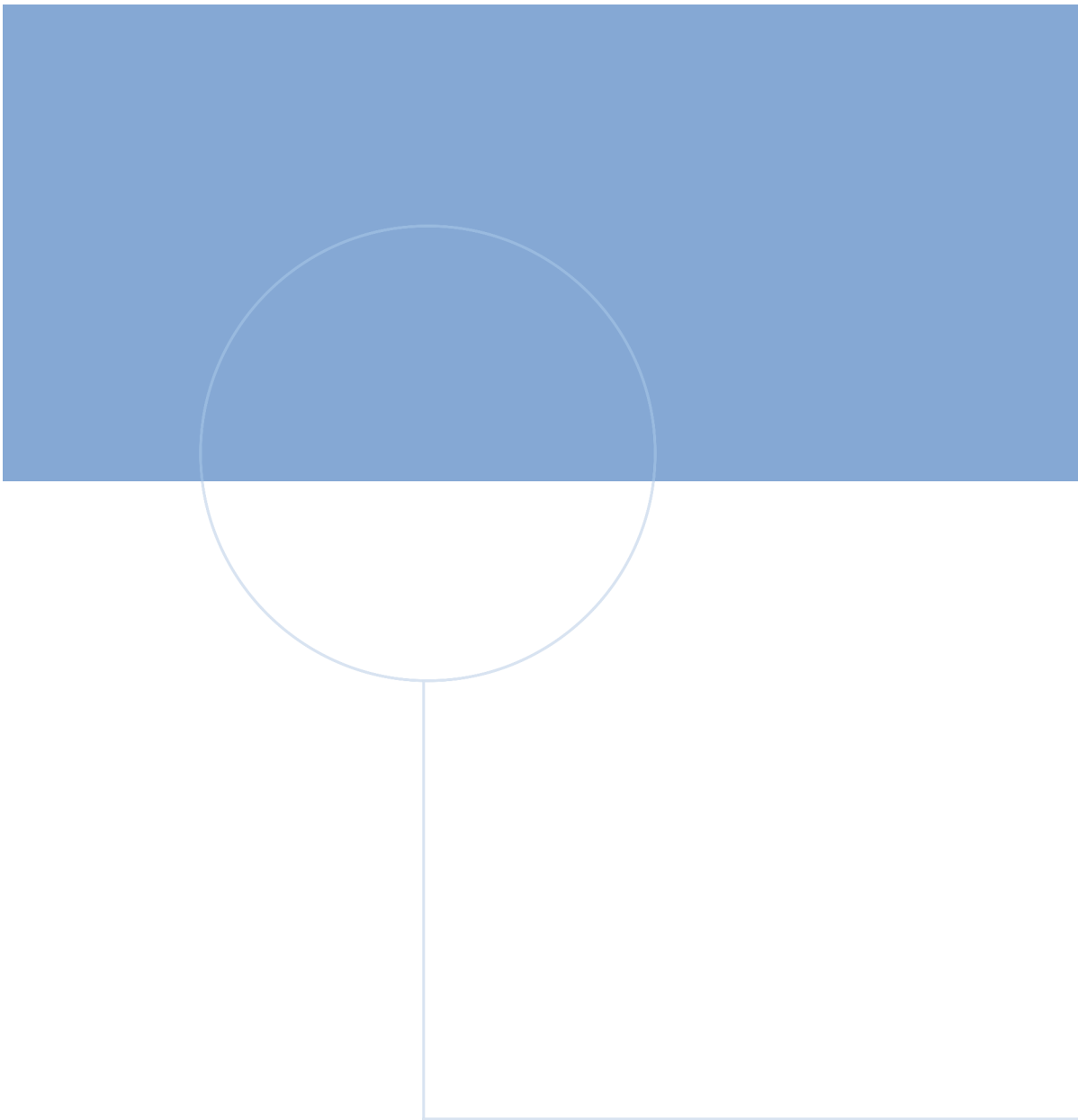
```
                                local pos_comp = cultsim.get_component(e,
                                component.position)
                                local boid_comp = cultsim.get_component(e,
                                component.lua1)

                                pos_comp.position.x = pos_comp.position.x +
                                (boid.speed * boid.velocity.x * dt)
                                pos_comp.position.y = pos_comp.position.y +
                                (boid.speed * boid.velocity.y * dt)
                        end)
                end
        }
}

-- Set system metatable for all systems
for k,v in pairs(systems) do
    setmetatable(v, {
        __index = cultlib.LuaSystem,
        __tostring = function(table) return k end
    })
end

--entities/boids.lua
entity = {
        PositionComponent = {
                position = Vec2:new(random:uniform(-scenario.bounds.x, scenario.bounds.x),
                                    random:uniform(-scenario.bounds.y, scenario.bounds.y))
        },
        SpriteComponent = {
                color = Vec3:new(random:uniform(0,255),random:uniform(0,255),
                                random:uniform(0,255)),
                texture = "sprites/arrow_c.png",
                normal = "sprites/arrow_n.png",
                scale = 10
        },
        VisionComponent = {
                radius = 50.0,
                fov = 0
        },
        TagComponent = {
                tags = ETag.Avoidable | ETag.Creature
        },
        LuaComponent = {
                lua_id = 1,
                speed = 80.0,
                velocity = Vec2:new(0,0)
```

```
        }
    }
```



NTNU

Norwegian University of
Science and Technology