МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**МОДЕЛИРОВАНИЕ И ИССЛЕДОВАНИЕ ПОВЕДЕНИЯ АГЕНТОВ С ПОМОЩЬЮ НЕЙРОЭВОЛЮЦИИ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.С. Прозоров

(подпись)

Направление подготовки 02.03.02 — «Фундаментальная информатика и\_\_\_\_\_

(код, наименование)

информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_курс\_\_\_\_\_\_\_\_3\_\_\_\_\_\_\_\_

Направленность (профиль) \_\_\_\_\_\_\_\_\_\_\_\_Фундаментальная информатика и информационные технологии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель

канд. техн. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Приходько

(подпись, дата)

Нормоконтролер

ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Нигодин

(подпись, дата)

Краснодар

2022

**РЕФЕРАТ**

Курсовая работа 26 стр., 3 ч., 9 рис., 5 источников, 4 приложения.

ИСКУССТВЕННАЯ ЖИЗНЬ, НЕЙРОННАЯ СЕТЬ, СИМУЛЯТОР, FLUTTER

Цель работы курсовой работы – создать модель искусственной жизни и разработать симулятор для этой модели.

Объектом исследования в работе является модели искусственной жизни.

Предметом исследования являются алгоритмы моделирования искусственной жизни.

В качестве методов исследования использовались эмпирический метод, абстрагирование, наблюдение, моделирование и эксперименты. В результате работы были изучены принципы моделирования искусственной жизни.

Научная новизна работы заключается в формировании новой модели искусственной жизни и разработке симулятора для изучения это модели.

По результатам работы, был разработан кроссплатформенный симулятор с визуализацией и пользовательским интерфейсом, предназначенный для моделирования и исследования поведения агентов.

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc104396660)

[1 Теоретическая информация о искусственной жизне 6](#_Toc104396661)

[1.1 Общие сведения 6](#_Toc104396662)

[1.2 Модели искусственной жизни 7](#_Toc104396663)

[1.3 Адаптивное поведение 12](#_Toc104396664)

[2 Искуссвтенные нейронные сети 15](#_Toc104396670)

[2.1 Общие сведения 15](#_Toc104396671)

[2.2 Биологический аспект 16](#_Toc104396672)

[2.3 Обучение нейронной сети 17](#_Toc104396673)

[2.4 Архитектура нейронной сети 18](#_Toc104396674)

[3 Реализация симулятора 20](#_Toc104396679)

[3.1 Инструменты для разработки 20](#_Toc104396680)

[3.2 Требования к реализации 20](#_Toc104396681)

[3.3 Принцип работы программы 20](#_Toc104396682)

[Заключение 26](#_Toc104396683)

[Список использованных источников 27](#_Toc104396684)

[Приложение A Файл инициализации программы 28](#_Toc104396685)

[Приложение Б Файл менеджера карты 30](#_Toc74084991)

[Приложение В Файл контроллера растений 35](#_Toc74084992)

[Приложение Г Файл контроллера агентов 38](#_Toc74084993)

# ВВЕДЕНИЕ

Актуальность данной работы состоит в том, что развитие технологий искусственной жизни приобретает более широкую значимость для современных исследований биологических систем, в особенности для обнаружения и изучения различных процессов, которые могут возникать в таких системах, а также для изучения природы воздействия эволюционных процессов на эти системы.

Искусственная жизнь – это дисциплина, которая была названа Кристофером Лэнгтоном, американским биологом-теоретиком. В ней исследователи изучают системы, связанные с естественной жизнью, ее процессами и ее эволюцией, с помощью моделирования с использованием компьютерных моделей, робототехники и биохимии. Искусственная жизнь применяется для моделирования процессов в экономике, поведения животных и насекомых, а также взаимодействия различных объектов. Искусственная жизнь изучает фундаментальные процессы живых систем в искусственных средах, чтобы получить более глубокое понимание сложной обработки информации, которая определяет такие системы.

Искусственная жизнь имеет дело с эволюцией агентов или популяций организмов, существующих лишь в виде компьютерных моделей, в искусственных условиях. Целью является изучение эволюции в реальном мире и возможности воздействия на её течение, например, с целью устранить некоторые наследственные ограничения. Разработчики, которые ведут исследования в данной сфере, надеются, что путем моделирования биологических систем можно лучше понять, почему и как они работают. С помощью моделей разработчики могут управлять созданной средой, проверять различные гипотезы и наблюдать, как системы и среда реагируют на изменения.

В данной курсовой работе рассматривается программная модель искусственной на основе нейронных сетей.

Помимо исследовательской работы, запланировано разработать симулятор, предназначенный для моделирования и исследования поведения агентов.

Основная цель работы – создать модель искусственной жизни и разработать симулятор для этой модели.

Для реализации поставленной цели предполагается решить следующие задачи:

* изучить принципы моделирования искусственной жизни;
* создать собственную модель искусственной жизни;
* выбрать инструментарий для разработки симулятора;
* разработать симулятор для модели искусственной жизни.

Объектом исследования в работе является модели искусственной жизни.

Предметом исследования являются алгоритмы моделирования искусственной жизни.

Информационная база исследования включает в себя несколько видов учебных материалов таких как статьи от авторитетных авторов, доклады с конференций и различные видеоматериалы с демонстрацией других моделей искусственной жизни. В качестве методов исследования использовались эмпирический метод, абстрагирование, наблюдение, моделирование и эксперименты.

Научная новизна работы заключается в формировании новой модели искусственной жизни и разработке симулятора для изучения это модели.

Теоретическая и практическая значимость работы состоит в потенциальном применении модели искусственной жизни. Например, подобные модели могут применяться в таких областях, как: искусственный интеллект, робототехника, медицина, нанотехнологии. Кроме того, схожие модели могут применяется при моделировании и исследовании социальных и экономических систем. Также, предложенная модель может использоваться для проведения исследований в области цифровой эволюции.

По результатам работы, был разработан кроссплатформенный симулятор с визуализацией и пользовательским интерфейсом, предназначенный для моделирования и исследования поведения агентов.

# Теоретическая информация о искусственной жизни

## 1.1 Общие сведения

В начале девяностых годов возникло одно интересное направление кибернетических исследований: «Искусственная жизнь» (англ. Artificial Life или Alife).

Основной мотивацией исследований искусственной жизни служит желание понять и промоделировать формальные принципы организации биологической жизни. Как сказал руководитель первой международной конференции по искусственной жизни К. Лангтон «основное предположение искусственной жизни состоит в том, что «логическая форма» организма может быть отделена от материальной основы его конструкции».

Сторонники направления «Искусственная жизнь» часто считают, что они исследуют более общие формы жизни, чем те, которые существуют на Земле. Т.е. изучается жизнь, какой она могла бы в принципе быть («life-as-it-could-be»), а не обязательно та жизнь, какой мы ее знаем («life-as-we-know-it»).

Искусственная жизнь [3] — это синтетическая биология, которая по аналогии с синтетической химией пытается воспроизвести биологическое поведение в различных средах. Это жизнь, созданная человеком, а не природой («life made by Man rather than by Nature»). Исследования искусственной жизни направлены не только на теоретические исследования свойств жизни, но и (аналогично синтетической химии) на практические приложения, такие как подвижные роботы, медицина, нанотехнология, моделирование социальных систем и тому подобное.

Большую роль в исследованиях искусственной жизни играет математическое и компьютерное моделирование. Очень часто «организмы» в искусственной жизни — это придуманные людьми организмы, живущие в мире компьютерных программ.

Отметим, хотя лозунг «Искусственная жизнь» был провозглашен в начале девяностых, в действительности идейно близкие модели разрабатывались и раньше.

В 60-х годах блестящий кибернетик и математик М. Л. Цетлин предложил и исследовал модели автоматов, способных адаптивно приспосабливаться к окружающей среде. Работы М. Л. Цетлина инициировали целое научное направление, получившее название "коллективное поведение автоматов" [4].

В 60-70-х годах под руководством талантливого кибернетика М. М. Бонгарда была построена весьма нетривиальная модель «Животное», характеризующая адаптивное поведение искусственных организмов, живущих на разбитой на клетки плоскости и обладающих рядом конкурирующих между собой потребностей [5].

## Модели искусственной жизни

Модели искусственной жизни – активно развивающаяся область исследований. Большинство моделей – остроумные компьютерные эксперименты. Серьезное математическое описание здесь только начинается [3].

Исследования искусственной жизни тесно связаны с другими интересными направлениями: моделями происхождения жизни, автоматами С.А. Кауффмана, работами по прикладному эволюционному моделированию, по теории нейронных сетей. Эволюция популяций искусственных организмов – одно из ведущих направлений исследований искусственной жизни. Модели эволюции здесь часто основаны на генетическом алгоритме. Но в моделях искусственной жизни часто не вводится явно понятие приспособленности, как это делается в генетических алгоритмах. Приспособленность проявляется естественным путем: особи рождаются, когда их предки готовы дать потомков, и погибают, когда не хватает пищи или когда их убивает другая особь. В этом случае говорят, что приспособленность эндогенна. Управление поведением искусственных организмов часто моделируется с помощью нейронных сетей. Модели искусственной жизни проливают новый свет на эволюционные явления.

Некоторые примеры характерных исследований искусственной жизни.

1) Исследование динамики жизнеподобных структур в клеточных автоматах (К.Лангтон) [3].

2) ПолиМир (PolyWorld) Л. Ягера: компьютерная модель искусственных организмов, которые имеют структурированную нейронную сеть, обладают цветовым зрением, могут двигаться, питаться (и увеличивать тем самым свою энергию), могут скрещиваться и бороться друг с другом [3]. Население обычно исчисляется сотнями, поскольку каждый индивидуум довольно сложен, а окружающая среда потребляет значительные компьютерные ресурсы. Было замечено, что после длительной эволюции спонтанно возникает ряд интересных форм поведения, таких как каннибализм, хищники и жертвы, а также мимикрия. На рисунке 1 изображён скриншот работы Polyworld.

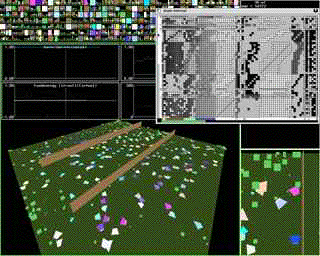


Рисунок 1 – Скриншот работы Polyworld

3) Тьерра (Tierra) Т.Рэя: модель эволюции самовоспроизводящися компьютерных программ [3]. «Организмы» Тьерры содержат геномы, которые определяют инструкции исполнительных программ. Есть операционная система, которая определяет: работу с памятью, выделение времени на вычисления каждому существу, подсчёт смертей и рождений, банк успешных геномов, автоматизирует экологический анализ. А программами этой операционной системы являются организмы. Взаимодействия между организмами проводят к эволюционному возникновению сложного «биоразнообразия» самовоспроизводящихся программ. На рисунке 2 изображён скриншот запущенной симуляции Tierra.



Рисунок 2 – Скриншот запущенной симуляции Tierra

4) Авида (Avida) К. Адами с сотрудниками , эта модель – развитие модели Тьерра [3]. По сравнению с Тьерой Авида более проста и обладает большей общностью. Модель анализировалась аналитическими методами. Были получены характеристики распределения особей в эволюционирующих популяциях. Исследования на модели Авида количественно поддерживают ту точку зрения, что эволюция движется скачками, а не непрерывно. На рисунке 3 изображён скриншот работы Avida.

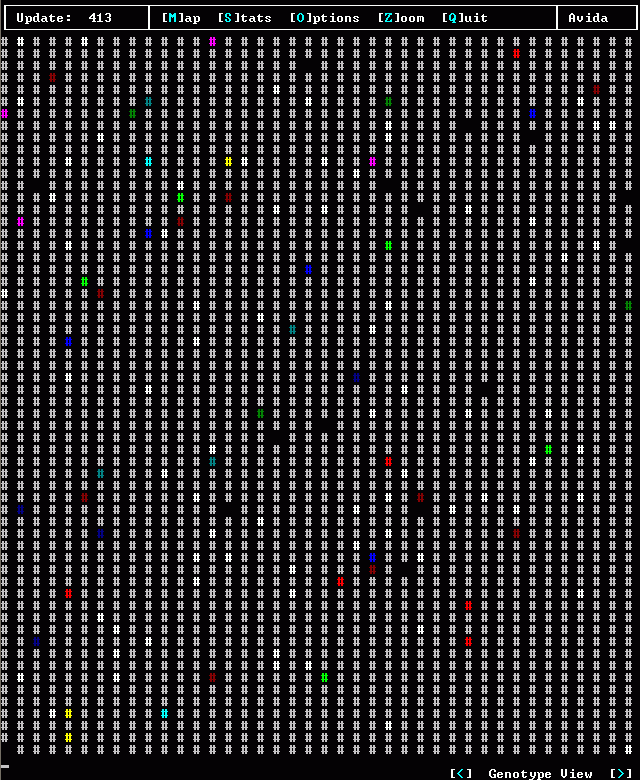


Рисунок 3 – Скриншот работы Avida с открытой вкладкой карты

5) Проект “Искусственная жизнь” foo52ru. В этом проекте не происходит прямого программирования алгоритма эволюции или генетического алгоритма, а он возникает автоматически за счёт среды. Используются агенты с программой в виде чисел для программирования поведения. Программа представляет собой ассемблеро-подобный язык, мутация может изменить случайную команду. Агентам необходимо собирать еду и выживать рядом друг с другом и под действием жестокой среды. На рисунке 4 изображён скриншот работы программы «Искусственная жизнь» foo52ru.

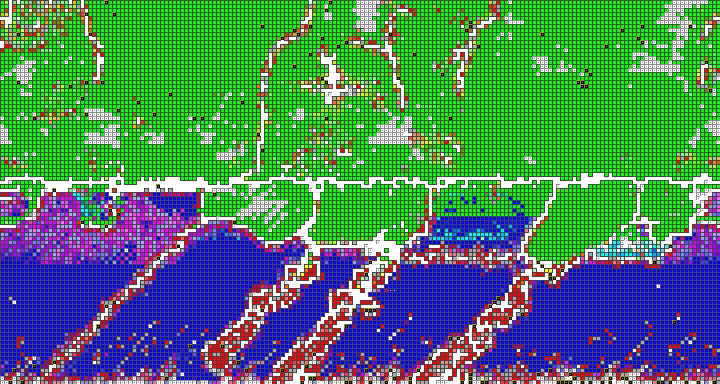


Рисунок 4 – Скриншот работы программы «Искусственная жизнь» foo52ru

6) Анализ взаимодействия между обучением и эволюцией, выполненный Д.Экли и М. Литтманом [3]. Эта работа продемонстрировала, что «обучение и эволюция вместе более успешны в формировании адаптивной популяции, чем обучение либо эволюция по отдельности».

7) ЭХО (ECHO) Дж.Холланда [3]. Эта модель описывает эволюцию простых агентов, которые взаимодействуют между собой путем скрещивания, борьбы и торговли. Взаимодействия между агентами приводит к формированию различных экологических систем: «войны миров», симбиозов и тому подобное.

8) Модель эволюции двух конкурирующих популяций, одна из которых есть популяция программ, решающих определенную прикладную проблему (задачу сортировки), а вторая – популяция задач, эволюционирующих в направлении усложнения проблемы (Д. Хиллис) [3]. Первая из популяций может рассматриваться как популяция особей-хозяев, а вторая как популяция паразитов. Моделирование показало, что коэволюция в системе паразит-хозяин проводит к нахождению значительно лучших решений проблемы по сравнению с тем решением, которое можно найти в результате эволюции одной только первой популяции (популяции особей-хозяев).

9) «Муравьиная ферма» (AntFarm) Р. Коллинза и Д Джефферсона. Эта модель разработана на базе «Коннекшен-машины» (Connection-Machine). Модель имитирует поведение поиска пищи в огромных эволюционирующих популяциях искусственных муравьев [3].

10) Классифицирующие системы Дж.Холланда с сотрудниками [3]. Это модель эволюции когнитивного процесса. Классифицирующая система есть система индуктивного вывода, которая основана на наборе логических правил. Система правил оптимизируется как посредством обучения, так и эволюционным методом. В процессе обучения меняются приоритеты использования правил (т.е. меняются коэффициенты, характеризующие силу правил). При обучении используется так называемый алгоритм «пожарной бригады»: при успехе поощряются не только те правила, которые непосредственно привели к успешному действию, но и те, которые были предшественниками успеха. Поиск новых правил осуществляется эволюционным методом.

## 1.3 Адаптивное поведение

С начала девяностых годов активно развивается направление «Адаптивное поведение». Основной подход этого направления – конструирование и исследование искусственных (в виде компьютерной программы или робота) «организмов», способных приспосабливаться к внешней среде.

Исследователи направления «Адаптивное поведение» стараются строить такие модели, которые применимы к описанию поведения как реального животного, так и искусственного организма.

Задачи направления «Адаптивное поведение»:

– исследовать архитектуры и принципы функционирования, которые позволяют животным или роботам жить и действовать в переменной внешней среде [3];

– попытаться проанализировать эволюцию когнитивных способностей животных и эволюционное происхождение человеческого интеллекта [3].

Для исследований «Адаптивное поведение» характерен синтетический подход: здесь конструируются архитектуры, обеспечивающие «Адаптивное поведение» поведение. Исследователь сам разрабатывает архитектуры, подразумевая, что подобные структуры, обеспечивающие адаптивное поведение, должны быть у реальных организмов.

Направления «Искусственная жизнь» и «Адаптивное поведение» имеют много общего: синтетический подход к конструированию жизнеподобных организмов, попытка промоделировать формальные законы жизни и систем управления, ориентация на компьютерные и математические модели, использование эволюционных концепций и моделей.

Эти направления используют ряд нетривиальных компьютерных методов:

– нейронные сети (англ. Neural Networks);

– классифицирующие системы (англ. Classifier Systems);

– обучение на основе подкрепления (англ. Reinforcement Learning);

– генетический алгоритм и другие методы эволюционной оптимизации.

В целом соотношение между направлениями «Искусственная Жизнь» и «Адаптивное Поведение», используемыми в них компьютерными методами, их научным значением и их потенциальными применениями можно представить в виде схемы, показанной на рисунке 5.

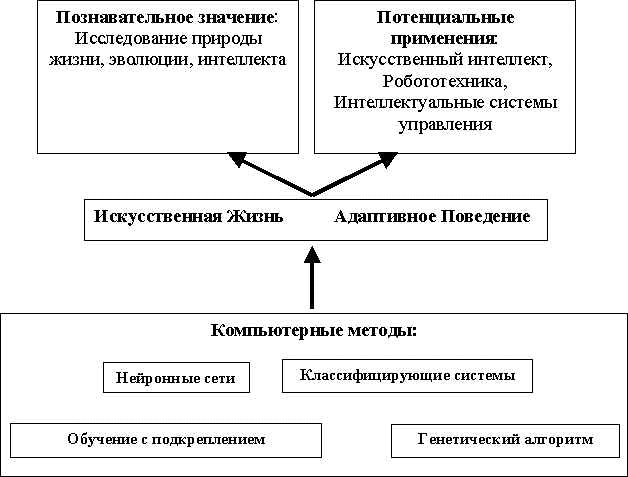


Рисунок 5 – Схема междисциплинарных связей направлений «Искусственная Жизнь» и «Адаптивное Поведение»

Это всё ещё активно развивающиеся направления исследований. По этим направлениям регулярно проводятся международные и европейские конференции «ALIFE (The Conference on Artificial Life)», «ECAL (European Conference on Artificial Life)». Издаётся журнал «Artificial Life».

# Искусственные нейронные сети

## 2.1 Общие сведения

Искусственные нейронные сети (ИНС) [1] – математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети Маккалока и Питтса. Впоследствии, после разработки алгоритмов обучения, получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

ИНС представляют собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты, особенно в сравнении с процессорами, используемыми в персональных компьютерах. Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т.п. С математической точки зрения, обучение нейронных сетей – это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования, нейронная сеть – способ решения проблемы эффективного параллелизма. А с точки зрения искусственного интеллекта, ИНС является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения – одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что, в случае успешного обучения, сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искаженных данных.

## 2.2 Биологический аспект

Нейронные сети возникли из исследований в области искусственного интеллекта, а именно, из попыток воспроизвести способность биологических нервных систем обучаться и исправлять ошибки, моделируя низкоуровневую структуру мозга (Patterson, 1996). Основной областью исследований по искусственному интеллекту в 60-е - 80-е годы были экспертные системы. Такие системы основывались на высокоуровневом моделировании процесса мышления (в частности, на представлении, что процесс нашего мышления построен на манипуляциях с символами). Скоро стало ясно, что подобные системы, хотя и могут принести пользу в некоторых областях, не ухватывают некоторые ключевые аспекты человеческого интеллекта. Согласно одной из точек зрения, причина этого состоит в том, что они не в состоянии воспроизвести структуру мозга. Чтобы создать искусственных интеллект, необходимо построить систему с похожей архитектурой.

Мозг состоит из очень большого числа нейронов, соединенных многочисленными связями. Нейроны – это специальная клетки, способные распространять электрохимические сигналы. Нейрон имеет разветвленную структуру ввода информации (дендриты), ядро и разветвляющийся выход (аксон). Аксоны клетки соединяются с дендритами других клеток с помощью синапсов. При активации нейрон посылает электрохимический сигнал по своему аксону. Через синапсы этот сигнал достигает других нейронов, которые могут в свою очередь активироваться. Нейрон активируется тогда, когда суммарный уровень сигналов, пришедших в его ядро из дендритов, превысит определенный уровень (порог активации) [2].

Интенсивность сигнала, получаемого нейроном, сильно зависит от активности синапсов. Каждый синапс имеет протяженность, и специальные химические вещества передают сигнал вдоль него. Один из самых авторитетных исследователей нейросистем, Дональд Хебб, высказал постулат, что обучение заключается в первую очередь в изменениях «силы» синаптических связей.

Таким образом, будучи построен из очень большого числа совсем простых элементов, каждый из которых берет взвешенную сумму входных сигналов и в случае, если суммарный вход превышает определенный уровень, передает дальше двоичный сигнал, мозг способен решать чрезвычайно сложные задачи.

## 2.3 Обучение нейронной сети

Процесс функционирования нейросети, то есть сущность действий, которые она способна выполнять, зависит от величин синаптических связей, поэтому, задавшись определенной структурой, отвечающей какой-либо задаче, разработчик сети должен найти оптимальные значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными).

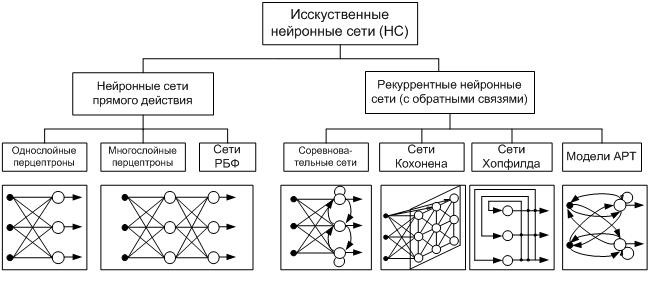
Этот этап называется обучением нейросети, и от того, насколько качественно он будет выполнен, зависит способность сети решать поставленные перед ней проблемы во время эксплуатации. На этапе обучения кроме параметра качества подбора весов важную роль играет время обучения. Как правило, эти два параметра связаны обратной зависимостью и их приходится выбирать на основе компромисса.

Обучение нейронной сети может вестись с учителем или без него. В первом случае сети предъявляются значения как входных, так и желательных выходных сигналов, и она по некоторому внутреннему алгоритму подстраивает веса своих синаптических связей. Во втором случае выходы нейросети формируются самостоятельно, а веса изменяются по алгоритму, учитывающему только входные и производные от них сигналы.

Существует великое множество различных алгоритмов обучения, которые делятся на два больших класса: детерминистские и стохастические. В первом из них подстройка весов представляет собой жесткую последовательность действий, во втором – она производится на основе действий, подчиняющихся некоторому случайному процессу.

## 2.4 Архитектура нейронной сети

ИНС может рассматриваться как направленный граф со взвешенными связями, в котором искусственные нейроны являются узлами. По архитектуре связей ИНС могут быть сгруппированы в два класса: сети прямого распространения, в которых графы не имеют петель, и рекуррентные сети, или сети с обратными связями.

Рисунок 6 – архитектуры ИНС

В наиболее распространенном семействе сетей первого класса, называемых многослойным перцептроном, нейроны расположены слоями и имеют однонаправленные связи между слоями. На рисунке 6 представлены типовые сети каждого класса. Сети прямого распространения являются статическими в том смысле, что на заданный вход они вырабатывают одну совокупность выходных значений, не зависящих от предыдущего состояния сети. Рекуррентные сети являются динамическими, так как в силу обратных связей в них модифицируются входы нейронов, что приводит к изменению состояния сети.

# Реализация симулятора

## 3.1 Инструменты для разработки

Для разработки был использован язык Dart 2.18.5, фреймворк Flutter 3.3.9. Также были использованы:

– пакет Freezed для генерации дата-классов;

– GitHub для контроля версий;

– Riverpod для управления состоянием.

## 3.2 Требования к реализации

Для лучшего контроля над симуляцией необходим пользовательский интерфейс с элементами управления. Также для лучшего и быстрого восприятия работы симуляции необходимо визуализировать ходы работы симуляции и предоставить интерактивную карту для более точного наблюдения. Поэтому симулятор должен:

– содержать элементы управления для запуска/остановки симуляции;

– содержать элементы управления для точно настройки количества шагов в секунду и количества отображаемых шагов в секунду;

– иметь статистику по симуляции, включая время начала симуляции, количество шагов, количества агентов и растений и количество их энергии, соответственно;

– иметь интерактивную карту с возможность перемещения, приближения и отдаления.

## 3.3 Принцип работы программы

Рассмотрим основные структуры данных в программе.

Модель агента:

– идентификатор;

– дата создания;

– запас энергии;

– направление взгляда;

– мозг (нейронная сеть).

Нейронная сеть агента содержит следующие входные нейроны:

– количество энергии агента;

– количество энергии агента;

– направление взгляда;

– если перед агентов есть растение, количество его энергии, иначе – 0;

– координата x направления с наибольшим количеством энергии растений в зоне 5 на 5 клеток;

– координата y направления с наибольшим количеством энергии растений в зоне 5 на 5 клеток.

Выходные нейроны сети:

– количество энергии агента;

– направление;

– питание;

– размножение.

Архитектура сети – многослойный перцептрон с пятью входными нейронами, двумя скрытыми слоями по пять нейронов и четырьмя выходными нейронами. Все нейроны содержат значения от -1 до 1.

Изначально нейронная сеть создаётся со случайными весами. Также её можно скопировать, но копирование будет сопровождаться мутациями. Шансы мутаций можно настроить.

Мозг агента является обёрткой для нейронной сети и принимает модель входа, а отдаёт модель выхода нейронной сети, которые преобразуют данные для и из этой сети для более удобной работы с ней.

На рисунке 7 представлена UML диаграмма агента и нейронной сети.

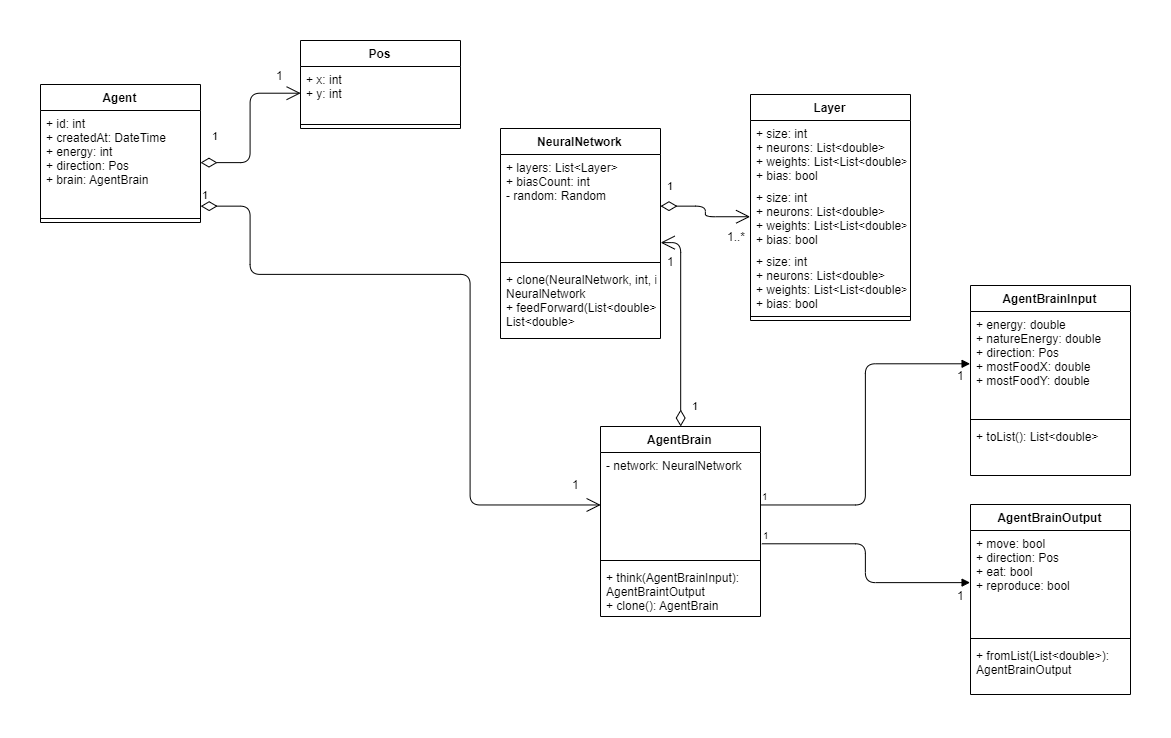


Рисунок 7 – UML диаграмма агента и нейронной сети

Модель растения:

– запас энергии.

Модель карты агентов:

– двумерный список агентов;

– размер карты.

Модель нужна для более удобной работы с агентами, так как переопределяет операторы доступа по индексу и возвращает агента по позиции на карте.

Модель карты растений:

– двумерный список растений;

– размер карты.

Модель нужна для более удобной работы с растениями, так как переопределяет операторы доступа по индексу и возвращает растение по позиции на карте.

Модель мировой карты:

– карта агентов;

– карта растений;

– размер карты;

– метод для проверки того, что позиция находится в пределах размеров карты.

Модель нужна для более удобной работы с предыдущими моделями карт.

Мир двумерный и содержит слой агентов и слой растений, слои разделены на клетки. Размер мира и количество начальных агентов и растений задаются в коде программы до начала симуляции.

За работу агентов отвечает класс AgentController. Он содержит метод для шага, который принимает карту мира и возвращает новую, в которой каждый агент делает свой ход. Именно в этом методе собираются входные данные для нейронной сети агента и на основе выходов совершается изменение направления взгляда, движение, питание и размножение, если всё это позволяет запас энергии. Если агент решать питаться растением рядом, то количество высчитывается коэффициент k по формуле (1):

. (1)

где

– количество энергии.

И агенту прибавляется количество энергии, которое высчитывается по формуле (2):

. (2)

где

– количество энергии у растения.

При этом у растения отнимается энергия, посчитанная по формуле (3):

. (3)

где

– количество энергии у растения.

При размножении в свободной клетке создаётся новый агент с новой нейронной сетью, которая является копией нейронной сети предка с определенными мутациями весов.

Похожий класс отвечает за работу растений – NatureController. Он содержит точно такой же метод, но тут алгоритм проще. Каждый шаг случайно выбирается пол процента пол процента клеток. Если на клетке есть растение, то оно совершает свой ход – выбирает клетку рядом и если там нет растений, то создаёт новое, а если есть, то у него берётся часть энергии. Также каждый ход растений получает случайное количество энергии – от 2 до 6. Если энергия растения оказывает меньше или равна нулю, то растение погибает. Этот алгоритм позволяет смоделировать распространение травы на карте.

На рисунке 8 представлена UML диаграмма моделей агентов, растений, карт и контроллеров.

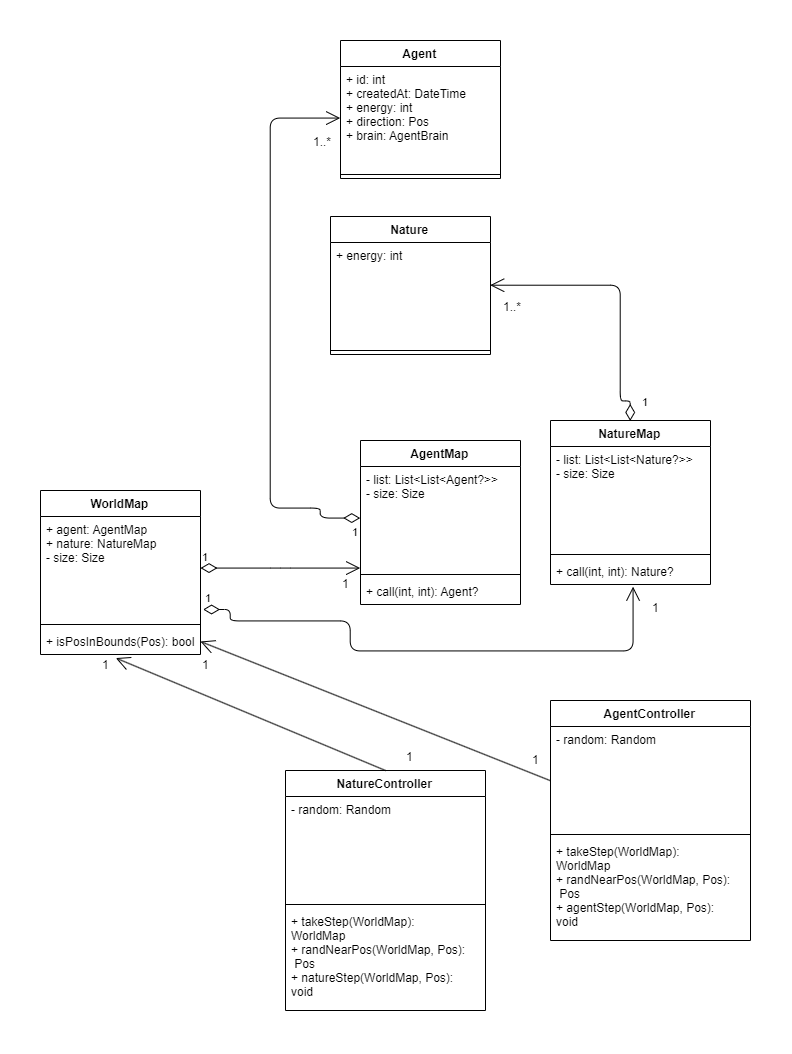


Рисунок 8 – UML диаграмма карты, контроллеров, моделей агентов и растений

Вся симуляция контролируется с помощью класса WorldMapManager. Этот класс отвечает за начальное расположение агентов и растений, а также за сбор всей статистики во время симуляции. Класс SimulationManager используется из пользовательского интерфейса для запуска симуляции и изменения её настроек. На рисунке 9 представлена UML диаграмма менеджеров и контроллеров.

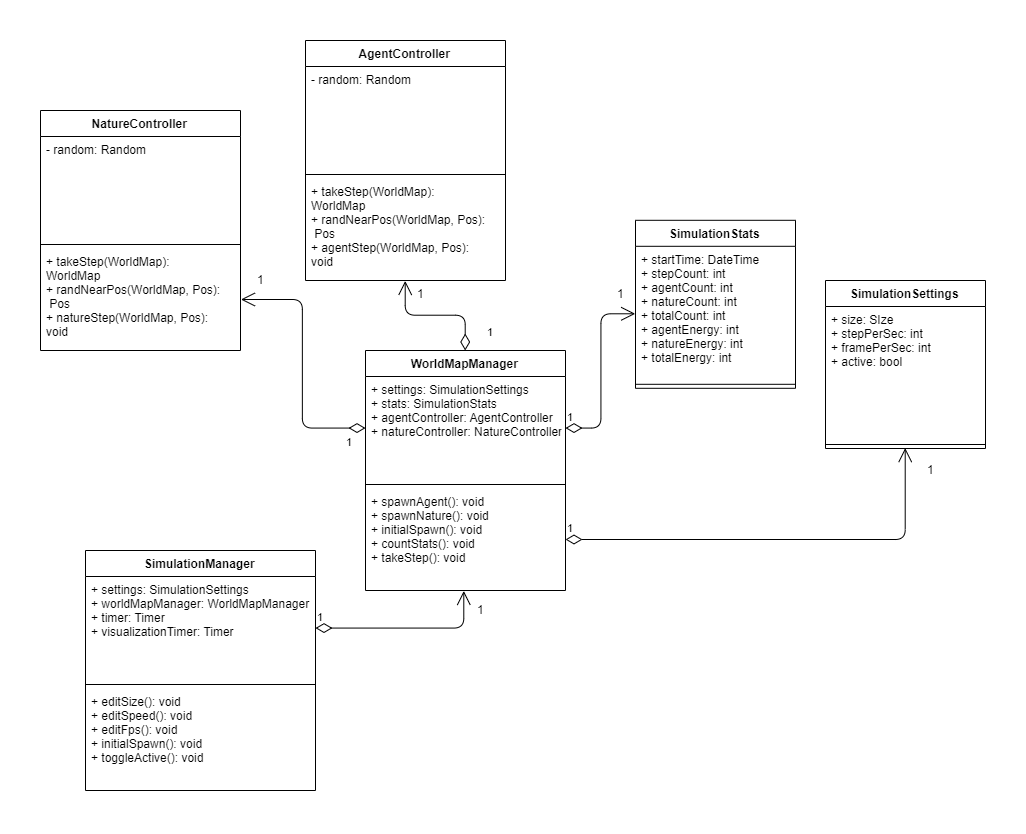


Рисунок 9 – UML диаграмма менеджеров и контроллеров

# ЗАКЛЮЧЕНИЕ

В работе были рассмотрены основные принципы моделирования искусственной жизни и принципы работы нейронных сетей. Были подробно изучены возможности языка программирования Dart и фреймворка Flutter.

В работе были поставлены и решены следующие задачи:

* изучены принципы моделирования искусственной жизни;
* на основе изученного материала построена собственную модель искусственной жизни;
* выбран инструментарий для разработки симулятора;
* разработан симулятор для модели искусственной жизни.

По результатам работы, был разработан кроссплатформенный симулятор с визуализацией и пользовательским интерфейсом, предназначенный для моделирования и исследования поведения агентов.

Однако, для прикладного применения модель пока не подходит, так как в ней рассмотрены только базовые принципы эволюции, и она всё ещё достаточно тривиальна. Также проблемой является подбор оптимальных настроек для симуляции, что сильно усложняет работу с моделью.

Модель можно дополнить различными системами, такими как: различные типы окружающей среды, система здоровья агентов, система производства и восприятия феромонов. Эти системы могут внести в модель большее разнообразие поведения и создадут больше различных интересных ситуаций.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 **Галушкин, А. И.** Нейронные сети: основы теории / А. И. Галушкин. - Москва : Горячая линия-Телеком, 2010. - 496 с. : ил., табл.; 22 см.; ISBN 978-5-9912-0082-0. — Текст : непосредственный.

2 **Каллан, Р.** Основные концепции нейронных сетей / Роберт Каллан; [Пер. с англ. А.Г. Сивака]. - М. [и др.] : Вильямс, 2001. - 287 с. : ил., табл.; 20 см.; ISBN 5-8459-0210-X. — Текст : непосредственный.

3 **Редько, В. Г.** Эволюционная кибернетика / В. Г. Редько. — М. : Наука, 2001. - 155, [1] с. : ил.; 21 см. — (Серия "Кибернетика: неограниченные возможности и возможные ограничения" / Рос. акад. наук).; ISBN 5-02-013063-x. — Текст : непосредственный.

4 **Цетлин, М. Л.** Исследования по теории автоматов и моделированию биологических систем / М. Л. Цетлин. - Москва : Наука, 1969. - 316 с., 1 л. портр. : ил.; 20 см. — Текст : непосредственный.

5 Моделирование обучения и поведения / АН СССР, Ин-т проблем передачи информации ; [Отв. ред. канд. физ.-мат. наук М. С. Смирнов]. - Москва : Наука, 1975. - 238 с. : ил.; 21 см. — Текст : непосредственный.

# ПРИЛОЖЕНИЕ A

**Файл инициализации программы**

Файл sim\_evo\lib\main.dart

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'features/app\_paint/app\_paint.dart';

import 'features/side\_panel/view/side\_panel.dart';

void main() {

runApp(

const ProviderScope(

child: MyApp(),

),

);

}

class MyApp extends StatelessWidget {

const MyApp({super.key});

@override

Widget build(BuildContext context) {

return const MaterialApp(

home: HomePage(),

);

}

}

class HomePage extends ConsumerWidget {

const HomePage({super.key});

@override

Widget build(BuildContext context, WidgetRef ref) {

return Scaffold(

body: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

Expanded(

child: InteractiveViewer(

maxScale: 50,

minScale: 0.1,

constrained: true,

child: const Padding(

padding: EdgeInsets.all(8),

child: AppPaint(),

),

),

),

const SidePanel(),

],

),

);

}

}

# ПРИЛОЖЕНИЕ Б

**Файл менеджера карты**

Файл sim\_evo\lib\managers\world\_map\_manager.dart

import 'dart:math';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import '../data/data\_models/agent/agent.dart';

import '../data/data\_models/nature/nature.dart';

import '../data/data\_models/pos/pos.dart';

import '../data/data\_models/simulation\_settings/simulation\_settings.dart';

import '../data/data\_models/simulation\_stats/simulation\_stats.dart';

import '../data/models/world\_map/world\_map.dart';

import '../features/agent\_brain/agent\_brain.dart';

import '../state\_holders/simulation\_settings.dart';

import '../state\_holders/simulation\_stats.dart';

import '../state\_holders/world\_map.dart';

import 'controllers/agent\_controller.dart';

import 'controllers/nature\_controller.dart';

final worldMapManager = Provider<WorldMapManager>((ref) {

return WorldMapManager(

settingsNotifier: ref.watch(simulationSettings.notifier),

statsNotifier: ref.watch(simulationStats.notifier),

worldMapNotifier: ref.watch(worldMap.notifier),

);

});

class WorldMapManager {

SimulationSettingsNotifier settingsNotifier;

SimulationStatsNotifier statsNotifier;

WorldMapNotifier worldMapNotifier;

SimulationSettings get settings => settingsNotifier.state;

SimulationStats get stats => statsNotifier.state;

WorldMap get worldMap => worldMapNotifier.state;

AgentController agentController = AgentController();

NatureController natureController = NatureController();

WorldMapManager({

required this.settingsNotifier,

required this.statsNotifier,

required this.worldMapNotifier,

});

void spawnAgent() {

final height = worldMap.size.height.toInt();

final width = worldMap.size.width.toInt();

final size = height \* width;

for (int i = 0; i < 0.001 \* size; i++) {

final pos = Pos(Random().nextInt(width), Random().nextInt(height));

if (worldMap.agent[pos] == null) {

worldMap.agent[pos] = Agent(

id: 0,

createdAt: DateTime.now(),

energy: Random().nextInt(4000),

direction: Pos(0, 0),

brain: AgentBrain(),

);

}

}

}

void spawnNature() {

final height = worldMap.size.height.toInt();

final width = worldMap.size.width.toInt();

final size = height \* width;

for (int i = 0; i < 0.005 \* size; i++) {

final pos = Pos(Random().nextInt(width), Random().nextInt(height));

worldMap.nature[pos] = Nature(energy: 30);

}

}

void initialSpawn() {

spawnAgent();

spawnNature();

statsNotifier.edit(

stats.copyWith(

startTime: DateTime.now(),

),

);

}

void countStats() {

int agentCount = 0;

int agentEnergy = 0;

int natureCount = 0;

int natureEnergy = 0;

for (int i = 0; i < worldMap.size.height; i++) {

for (int j = 0; j < worldMap.size.width; j++) {

if (worldMap.agent(i, j) != null) {

agentCount++;

agentEnergy += worldMap.agent(i, j)!.energy;

} else if (worldMap.nature(i, j) != null) {

natureCount++;

natureEnergy += worldMap.nature(i, j)!.energy;

}

}

}

// if (agentCount < 50) {

// spawnAgent();

// }

if (stats.stepCount % 100 == 0) {

print('$agentCount ! $natureCount');

print('${stats.agentEnergy} ? ${stats.natureEnergy}');

}

statsNotifier.edit(

stats.copyWith(

stepCount: stats.stepCount + 1,

agentCount: agentCount,

agentEnergy: agentEnergy,

natureCount: natureCount,

natureEnergy: natureEnergy,

),

);

}

void takeStep() {

natureController.takeStep(worldMap);

agentController.takeStep(worldMap);

countStats();

}

}

# ПРИЛОЖЕНИЕ В

**Файл контроллера растений**

Файл sim\_evo\lib\managers\controllers\nature\_controller.dart

import 'dart:math';

import '../../data/data\_models/nature/nature.dart';

import '../../data/data\_models/pos/pos.dart';

import '../../data/models/world\_map/world\_map.dart';

class NatureController {

final \_random = Random();

WorldMap takeStep(WorldMap map) {

// for (var i = 0; i < map.size.height; i++) {

// for (var j = 0; j < map.size.width; j++) {

// final pos = Pos(j, i);

// if (map.nature[pos] != null) {

// natureStep(map, pos);

// }

// }

// }

final width = map.size.width.toInt();

final height = map.size.height.toInt();

final count = 0.005 \* width \* height;

for (int i = 0; i < count; i++) {

final pos = Pos(

\_random.nextInt(width),

\_random.nextInt(height),

);

if (map.nature[pos] != null) {

natureStep(map, pos);

}

}

return map;

}

Pos randNearPos(WorldMap map, Pos pos) {

final x = pos.x + \_random.nextInt(3) - 1;

final y = pos.y + \_random.nextInt(3) - 1;

if (x >= 0 &&

x < map.size.width &&

y >= 0 &&

y < map.size.height &&

x != 0 &&

y != 0) {

return Pos(x, y);

} else {

return randNearPos(map, pos);

}

}

void natureStep(WorldMap map, Pos pos) {

final nature = map.nature[pos]!;

// if (\_random.nextInt(100) < 1) {

final nearPos = randNearPos(map, pos);

if (map.nature[nearPos] == null) {

map.nature[nearPos] = Nature(energy: 5);

} else {

final e = (nature.energy \* 0.1).toInt();

final otherNature = map.nature[nearPos]!;

if (otherNature.energy + e < 100) {

map.nature[nearPos]!.energy += e;

}

}

// }

// if (\_random.nextInt(100) < 5) {

if (nature.energy < 100) {

nature.energy += \_random.nextInt(5) + 2;

}

//}

if (nature.energy <= 0) {

map.nature[pos] = null;

}

}

}

# ПРИЛОЖЕНИЕ Г

**Файл контроллера агентов**

Файл sim\_evo\lib\managers\controllers\agent\_controller.dart

import 'dart:math';

import '../../data/data\_models/agent/agent.dart';

import '../../data/data\_models/pos/pos.dart';

import '../../data/models/world\_map/world\_map.dart';

import '../../features/agent\_brain/agent\_brain.dart';

class AgentController {

final \_random = Random();

WorldMap takeStep(WorldMap map) {

for (var i = 0; i < map.size.height; i++) {

for (var j = 0; j < map.size.width; j++) {

final pos = Pos(j, i);

if (map.agent[pos] != null) {

agentStep(map, pos);

}

}

}

return map;

}

Pos randNearPos(WorldMap map, Pos pos) {

final x = pos.x + \_random.nextInt(3) - 1;

final y = pos.y + \_random.nextInt(3) - 1;

if (x >= 0 &&

x < map.size.width &&

y >= 0 &&

y < map.size.height &&

x != 0 &&

y != 0) {

return Pos(x, y);

} else {

return randNearPos(map, pos);

}

}

void agentStep(WorldMap map, Pos pos) {

final agent = map.agent[pos]!;

agent.energy--;

if (agent.energy <= 0) {

map.agent[pos] = null;

return;

}

final forwardPos = pos + agent.direction;

final posNorm = map.isPosInBounds(forwardPos);

double nat = 0;

if (posNorm && map.nature[forwardPos] != null) {

nat = map.nature[forwardPos]!.energy / 100.0;

}

double mostFoodX = 0;

double mostFoodY = 0;

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

final x = pos.x + i - 4;

final y = pos.y + j - 4;

final ps = Pos(x, y);

if (map.isPosInBounds(ps)) {

if (map.nature[ps] != null) {

final e = map.nature[ps]!.energy / 100.0;

mostFoodX += x \* e;

mostFoodY += y \* e;

}

}

}

}

final input = AgentBrainInput(

energy: agent.energy / 100.0,

natureEnergy: nat,

direction: agent.direction,

mostFoodX: mostFoodX,

mostFoodY: mostFoodY,

);

final output = agent.brain.think(input);

agent.direction = output.direction;

if (agent.energy > 75) {

final nearPos = randNearPos(map, pos);

if (map.agent[nearPos] == null) {

map.agent[nearPos] = Agent(

id: 0,

createdAt: DateTime.now(),

energy: 20,

direction: agent.direction,

brain: agent.brain.clone(),

);

agent.energy -= 50;

}

}

if (posNorm && map.nature[forwardPos] != null) {

final natE = map.nature[forwardPos]!.energy;

final k = pow((natE > 0 ? natE : 0) / 100.0 + 0.02, 1 / 4) - 0.2;

//final k = ((natE > 0 ? natE : 0) / 100) \* 0.5 + 0.25;

final e = (natE).toInt();

if (agent.energy + e <= 100) {

agent.energy += (e \* k).toInt();

map.nature[forwardPos]!.energy -= e;

}

}

if (posNorm && output.move) {

if (posNorm && map.agent[forwardPos] == null) {

map.agent[forwardPos] = agent;

map.agent[pos] = null;

}

agent.energy -= 1;

}

}

}