

Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе №5 (Многоагентные системы)

Дисциплина: Интеллектуальные системы

Выполнил студент группы 13541/3

_____ А.Р. Раскин
(подпись)

Преподаватель

_____ Е.Н. Бендерская
(подпись)

Санкт-Петербург
2017 г.

Содержание

1	Лабораторная работа №5	2
1.1	Дайте определение и краткую классификацию многоагентных интеллектуальных систем. В чем преимущества и в чем недостатки многоагентного подхода?	2
1.1.1	Приведите наиболее полную классификацию таких систем, кратко поясните по какому признаку дана эта классификация.	4
1.1.2	Какие задачи решаются с помощью многоагентного подхода. Приведите не менее ДВУХ (2) примеров задач, с кратким описанием (желательно сопроводить рисунками, диаграммами для наглядности)	6
1.1.3	Проанализируйте преимущества и недостатки многоагентного подхода на примере задач п.1.3. Сформулируйте общие проблемы недостатки и общие преимущества достоинства такого подхода.	8
1.2	Практическая часть. Обзор Netlogo.	9
1.2.1	Процесс установки ПО, ссылки на сайты ссылки на необходимые драйвера и т.д. (если нужно)	9
1.2.2	Процесс создания простого проекта	11
1.2.3	Создание кнопки	12
1.3	Выводы	20

Лабораторная работа №5

1.1 Дайте определение и краткую классификацию многоагентных интеллектуальных систем. В чем преимущества и в чем недостатки многоагентного подхода?

Многоагентные системы или мультиагентные системы – это направление искусственного интеллекта, которое для решения сложной задачи или проблемы использует системы, состоящие из множества взаимодействующих агентов [mac1]. Считается, что один агент владеет всего лишь частичным представлением о глобальной проблеме, а значит, он может решить лишь некоторую часть общей задачи.

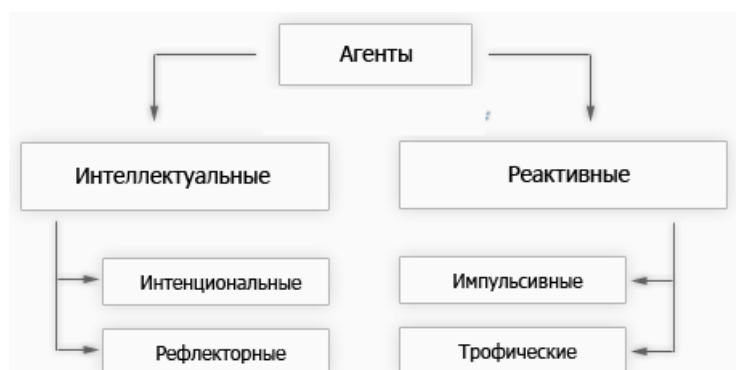


Рис. 1.1: Классификация.

Интеллектуальные агенты обладают хорошо развитой и пополняемой символьной моделью внешнего мира, что достигается благодаря наличию у них базы знаний, механизмов решения и анализа действий. Небольшое различие между этими типами интеллектуальных агентов связано с расстановкой акцентов на тех или иных интеллектуальных функциях: либо на получении знаний о среде, либо на рассуждениях о возможных действиях. У коммуникативных агентов внутренняя модель мира превращается главным образом в модель общения, состоящую из моделей участников, процесса и желаемого результата общения. Наконец, база знаний ресурсного агента содержит в основном знания о структуре и состоянии ресурсов, определяющих различные формы поведения.

У полноценного интеллектуального агента обязательно должны присутствовать как минимум четыре перечисленных функции: когнитивная, рассуждающая (а, в более общем контексте, регулятивная), коммуникативная и ресурсная.

Реактивные агенты не имеют ни сколько-нибудь развитого представления внешней среды, ни механизма многошаговых рассуждений, ни достаточного количества собственных ресурсов. Отсюда вытекает еще одно существенное различие между интеллектуальными и реактивными агентами, связанное с возможностями прогнозирования изменений внешней среды и, как следствие, своего будущего. В силу вышеуказанных недостатков реактивные агенты обладают очень ограниченным диапазоном предвидения. Они практически не способны планировать свои действия, поскольку реактивность в чистом виде означает такую структуру обратной связи, которая не содержит механизмов прогноза. Тогда как интеллектуальные агенты, благодаря богатым внутренним представлениям внешней среды и возможностям рассуждений, могут запоминать и анализировать различные ситуации, предвидеть возможные реакции на свои действия, делать из этого выводы, полезные для дальнейших действий и, в результате, планировать свое поведение. Именно развитые когнитивные и делиберативные способности позволяют таким агентам строить виртуальные миры, работая в которых они формируют планы действий.

Интеллектуальные агенты, будучи значительно автономнее реактивных, имеют куда ярче выраженную индивидуальность и характеризуются целесообразным поведением в сообществе агентов, а также стремлением использовать ресурсы других агентов для достижения собственных целей. В то же время, реактивные агенты, как это видно из самого их названия, работают в основном на уровне стимульно-реактивных связей, обладая очень бедной индивидуальностью и сильной зависимостью от внешней среды (сообщества агентов). Результаты сравнительного анализа реактивных и когнитивных агентов представлены в таблице. Интеллектуальные агенты, будучи значительно автономнее реактивных, имеют куда ярче выраженную индивидуальность и характеризуются целесообразным поведением в сообществе агентов, а также стремлением использовать ресурсы других агентов для достижения собственных целей. В то же время, реактивные агенты, как это видно из самого их названия, работают в основном на уровне стимульно-реактивных связей, обладая очень бедной индивидуальностью и сильной зависимостью от внешней среды (сообщества агентов). Результаты сравнительного анализа реактивных и когнитивных агентов представлены в таблице (рис. 2)

Характеристики	Когнитивные агенты	Реактивные агенты
Внутренняя модель внешнего мира	Развитая	Примитивная
Рассуждения	Сложные и рефлексивные рассуждения	Простые одношаговые рассуждения
Мотивация	Развитая система мотивации, включающая убеждения, желания, намерения	Простейшие побуждения, связанные с выживанием
Память	Есть	Нет
Реакция	Медленная	Быстрая
Адаптивность	Малая	Высокая
Модульная архитектура	Есть	Нет
Состав многоагентной системы	Небольшое число автономных агентов	Большое число зависимых друг от друга агентов

Рис. 1.2: Сравнительный анализ агентов

Далее, по типу поведения интеллектуальные агенты делятся на интенциональных и рефлексорных, а реактивные – на побуждаемых (импульсивных) и трофических. Большинство интеллектуальных (когнитивных) агентов можно отнести к числу интенциональных.

Подобные агенты наделены собственными механизмами мотивации. Это означает, что в них так или иначе моделируются внутренние убеждения, желания, намерения и мотивы, порождающие цели, которые и определяют их действия.

В свою очередь, модульные или рефлексорные агенты не имеют внутренних источников мотивации и собственных целей, а их поведение характеризуется простейшими (одношаговыми) выводами или автоматизмами.

1.1.1 Приведите наиболее полную классификацию таких систем, кратко поясните по какому признаку дана эта классификация.

Характеристики	Типы агентов			
	Простые	Смышленные (smart)	Интеллектуальные (intelligent)	Действительно интеллектуальные (truly)
Автономное выполнение	+		+	+
Взаимодействие с другими агентами и/или пользователями	+	+	+	+
Слежение за окружением	+	+	+	+
Способность использования абстракций		+	+	+
Способность использования предметных знаний		+	+	
Возможность адаптивного поведения для достижения целей			+	+
Обучение из окружения			+	+
Толерантность к ошибкам и/или неверным входным сигналам			+	
Real-time исполнения			+	
ЕЯ-взаимодействие			+	

Рис. 1.3: Классификации агентов.

Как следует из приведенной таблицы (информация взята с [4]), собственно целесообразное поведение появляется только на уровне интеллектуальных агентов. Для него необходимо не только наличие целей функционирования, но и возможность использования достаточно сложных знаний о среде, партнерах и о себе. Под агентом понимается аппаратная или программная сущность, способная действовать в интересах достижения целей, поставленных перед ним владельцем или пользователем. Свойства агента (объекта) описываются исходной системой, а правила поведения – порождающей системой. Состояние объекта определяется перечнем его свойств с текущими значениями[3].

1.1.2 Какие задачи решаются с помощью многоагентного подхода. Приведите не менее ДВУХ (2) примеров задач, с кратким описанием (желательно сопроводить рисунками, диаграммами для наглядности)

Многоагентные системы применяются в нашей жизни в графических приложениях, например, в компьютерных играх (все NPC, охарактеризованные как ИИ). Агентные системы также были использованы в фильмах (при создании спец-эффектов). Теория МАС используется в составных системах обороны. Также МАС применяются в транспорте, логистике, графике, геоинформационных системах, робототехнике и многих других. Многоагентные системы хорошо зарекомендовали себя в сфере сетевых и мобильных технологий, для обеспечения автоматического и динамического баланса нагрузки, расширяемости и способности к самовосстановлению.[1]

Рассмотрим более конкретные примеры.

Пример 1. Типичная задача электронной коммерции, в которой участвуют агенты-продавцы и агенты-покупатели (Рис. 1.4). Торговля осуществляется в электронном магазине, который представляет собой программу, размещенную на сервере. Ее основным назначением является организация взаимодействия агентов, интересы которых совпадают. Агенты действуют по поручению своих персональных пользователей. При этом агенты-продавцы стремятся продать свой товар по максимально возможной цене, а агенты-покупатели стремятся купить нужный товар по минимальной цене. Оба вида агентов действуют автономно и не имеют целей кооперации. Электронный магазин регистрирует появление и исчезновение агентов и организует контакты между ними, делая их «видимыми» друг для друга.

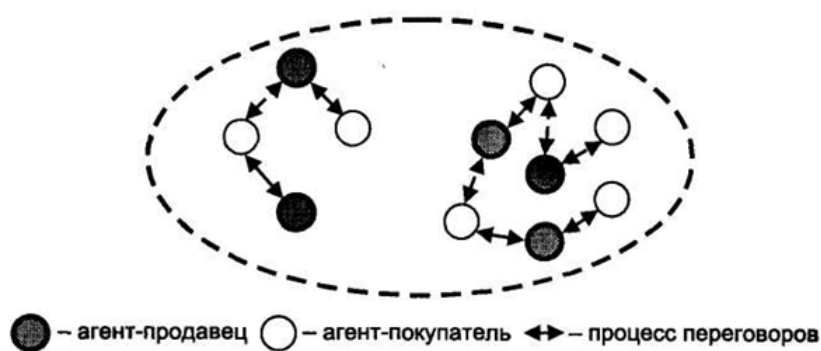


Рис. 1.4: Схема электронного магазина.

Поведение агента-продавца характеризуется следующими параметрами:

- желаемая дата, до наступления которой необходимо продать товар;
- желаемая цена, по которой пользователь хочет продать товар;
- самая низкая допустимая цена, ниже которой товар не продается;

- функция снижения цены во времени (линейная, квадратичная и др.);
- описание продаваемого товара.

Агент-покупатель имеет «симметричные» параметры:

- крайний срок покупки товара;
- желаемая цена покупки;
- самая высокая приемлемая цена;
- функция роста цены во времени;
- описание покупаемого товара.

Торги ведутся по схеме закрытого аукциона первой цены. Поведение агентов описывается простой моделью, в которой не используются знания и рассуждения. Агент-продавец, получив от электронного магазина информацию о потенциальных покупателях своего товара, последовательно опрашивает их всех с целью принять решение о возможности совершения сделки. Сделка заключается с первым агентом-покупателем, который готов дать за товар запрашиваемую цену. Продавец не может вторично вступить в контакт с любым покупателем до тех пор, пока не опросит всех потенциальных покупателей. При каждом контакте агент-продавец ведет переговоры, предлагая начальную цену либо снижая ее. Агент-покупатель действует аналогичным образом, отыскивая продавцов нужного товара и предлагая им свою цену покупки, которую он может увеличить в процессе переговоров. Любая сделка завершается только в случае ее одобрения пользователем агента.

Данная схема переговоров представляет собой простейший случай взаимодействия автономных агентов, действующих реактивно. Тем не менее итоговое поведение системы вполне адекватно реальности.

Пример 2. Менеджер сервис-центра, готовящий ежемесячный баланс по отремонтированному оборудованию, обнаруживает расхождение между оформленными бланками заказов и имевшимся в начале месяца на складе запасными частями, например, по ассортименту или количеству.

Менеджер вызывает систему и дает формализованное описание проблемы. Анализируя сложившуюся проблемную ситуацию, система приходит к выводу, что причина либо в неверном заполнении бланков заказов на работы приемщиками центра или ошибка в только что внедренной программе. Для разрешения этой проблемы система активизирует создание рабочей группы, включающей менеджера, программиста, приемщиц и кладовщика, каждый из которых должен проверить свои действия (обратим внимание, что каждый - из своего подразделения).

Пусть в результате обнаруживается ошибка в программе подготовки отчетов, вызванная методикой учета товара на складе. Изменение этой методики, в свою очередь, должно быть согласовано с бухгалтером центра (и, возможно, с директором), а если изменения коснулись технологии заполнения бланка, менеджер и приемщицы должны быть заново обучены программистом. Это вызывает организацию новых временных рабочих групп, которые действуют до момента восстановления ситуации.

Вся эта процедура может потребовать как полного блокирования всех действий по оформлению заказ-нарядов, так и осуществляться в фоновом режиме. Рабочие группы, сформированные системой в процессе решения этой проблемы представлены на рисунке 1.5.

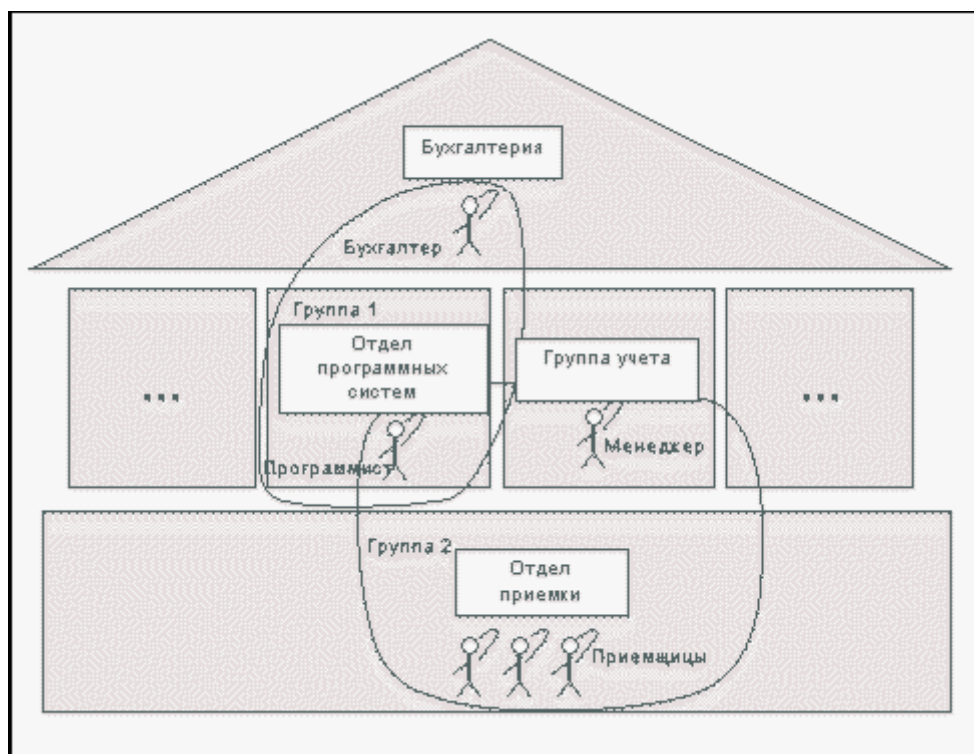


Рис. 1.5: Структура сервисного центра.

1.1.3 Проанализируйте преимущества и недостатки многоагентного подхода на примере задач п.1.3. Сформулируйте общие проблемы недостатки и общие преимущества достоинства такого подхода.

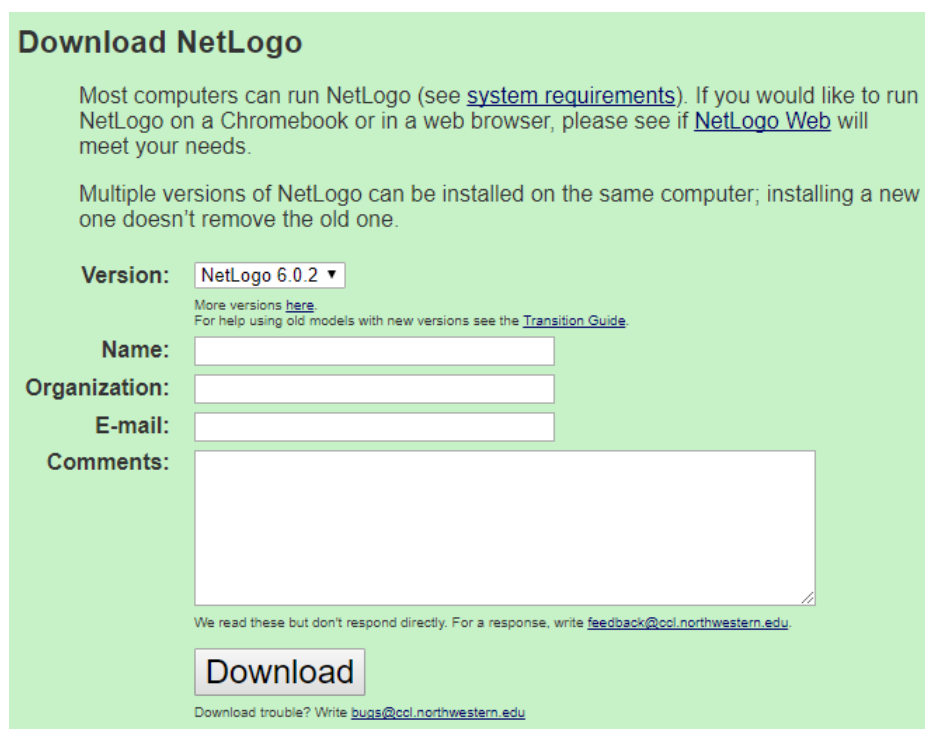
На приведенных примерах видно, как МАС способны к самоорганизации и самоуправлению: они способны решить, в какой момент какой сотрудник должен переключиться на другую задачу или наоборот, сильнее сконцентрироваться на текущей. При этом каждый агент обладает лишь той информацией, которая необходима ему для решения его собственных задач. При этом система сама так же рассчитывает необходимые убытки и прибыль и т.д.

1.2 Практическая часть. Обзор Netlogo.

NetLogo — агентно-ориентированный язык программирования и интегрированная среда разработки.

1.2.1 Процесс установки ПО, ссылки на сайты ссылки на необходимые драйвера и т.д. (если нужно)

Для того, чтобы скачать Netlogo необходимо перейти на сайт производителя (см. [2]).



The screenshot shows a web form titled "Download NetLogo" on a light green background. The form includes the following elements:





- Text:** "Most computers can run NetLogo (see [system requirements](#)). If you would like to run NetLogo on a Chromebook or in a web browser, please see if [NetLogo Web](#) will meet your needs."
- Text:** "Multiple versions of NetLogo can be installed on the same computer; installing a new one doesn't remove the old one."
- Version:** A dropdown menu currently showing "NetLogo 6.0.2". Below it, small text says "More versions [here](#). For help using old models with new versions see the [Transition Guide](#)".
- Name:** A text input field.
- Organization:** A text input field.
- E-mail:** A text input field.
- Comments:** A large text area for comments.
- Footer text:** "We read these but don't respond directly. For a response, write feedback@ccl.northwestern.edu."
- Download button:** A button labeled "Download".
- Footer text:** "Download trouble? Write bugs@ccl.northwestern.edu".

Рис. 1.6: Форма заполнения.

По желанию можно заполнить форму (или же просто оставить все поля пустыми). Жмем на кнопку «download», после этого станет доступен выбор среды для разных ОС. Среди них: Windows (x32, x64), Linux (x32, x64), Mac OS X. Т.к. на подопытном компьютере установлена Windows, были использованы исходники по Windows, далее процесс установку описан под ОС Windows.

NetLogo 6.0.2 Downloads

August 11, 2017

	Mac OS X	Download (191 MB)
	Windows (32-bit)	Download (171 MB)
	Windows (64-bit)	Download (173 MB)
	Linux (32-bit)	Download (192 MB)
	Linux (64-bit)	Download (190 MB)

[sign up for NetLogo community mailing lists](#)
(where many questions can be posted and answered)

Рис. 1.7: Выбор Системы.

Установка не совершенно не сложная. Никаких дополнительных настроек не требуется, достаточно просто нажимать кнопки «next». После завершения установки, появятся ярлыки среды.

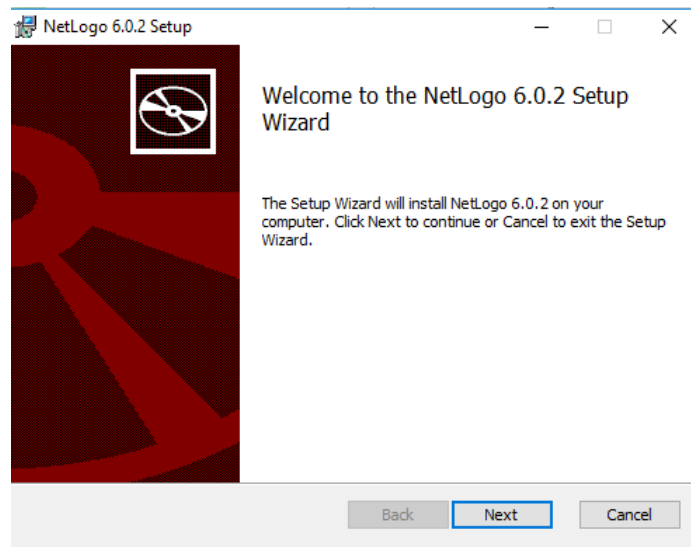


Рис. 1.8: Установка

1.2.2 Процесс создания простого проекта

На рисунке 1.9 представлено рабочее окно Netlogo.

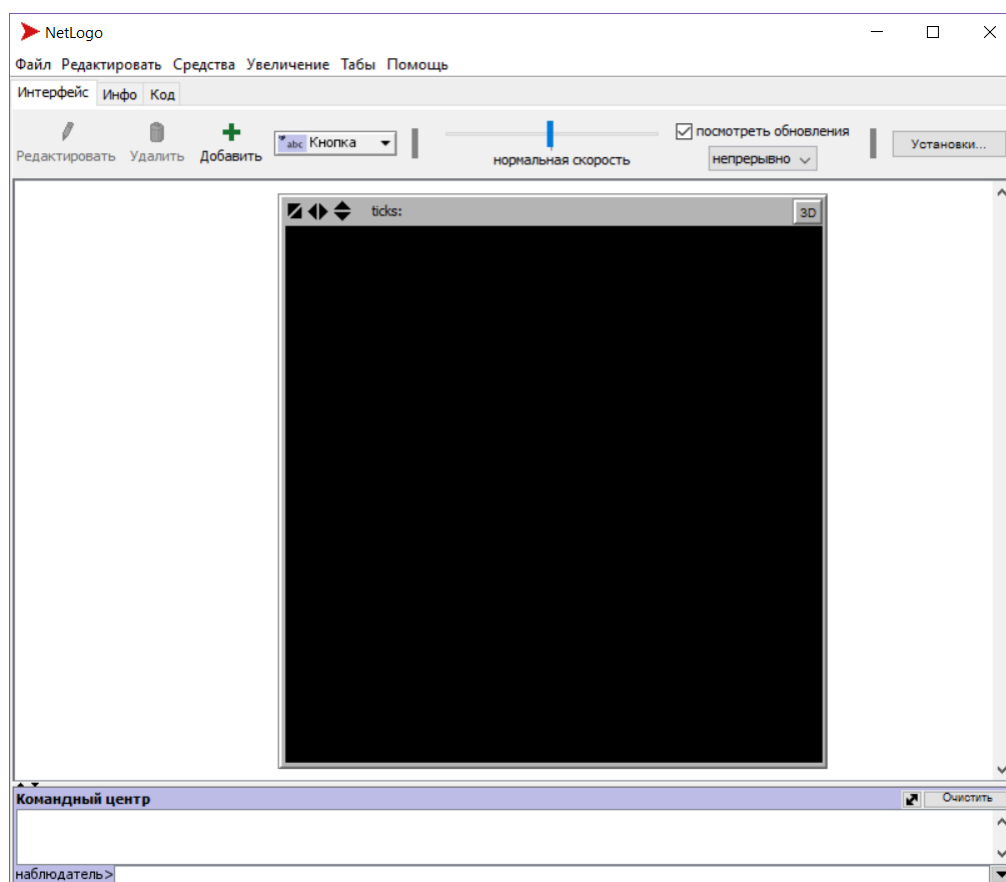


Рис. 1.9: Рабочее окно среды.

На самом верху окна расположено основное меню для работы с программой, состоящее из пунктов «Файл», «Редактировать», «Средства», «Увеличение», «Табы», «Помощь».

«Файл» — позволяет работать с файлами NetLogo: сохранять, открывать, переименовывать, создавать новые модели (файлы), открывать модели из встроенной библиотеки моделей NetLogo. Возле каждого из подпунктов можно увидеть комбинацию клавиш, которая позволит выполнить эту команду без открытия меню «Файл».

В меню «Файл» можно выбрать библиотеку моделей NetLogo. Открыв ее, в папке Sample Models можно найти наиболее тщательно протестированные и проверенные модели, из всех, что имеются в запасе NetLogo. Эти модели являются образчиком хорошего программирования и практической документации (объяснение назначения и подробное описание модели), которая всегда должна прилагаться к модели. В папке Code Examples находятся незаконченные модели, которые представляют собой короткие иллюстрации, конкретных функций NetLogo или техник программирования.

«Средства» — позволяет пользоваться инструментами, встроенными в программу. Такими инструментами являются, например, окна отслеживания глобальных переменных, объявленных в программе, связей, черепашек и патчей — агентов NetLogo. «Средства» также позволяет открывать диалог выбора цвета, где показан весь цветовой диапазон в NetLogo, переключаться в 3D измерение.

«Табы» — позволяет переключаться между вкладками «Интерфейс», «Инфо» и «Код», которые можно так же увидеть сверху под главным меню.

Вкладка «Код» отображает программный код, сопровождающий каждую модель. Данный код описывает действие того или иного инструмента, агентов и их взаимодействие.

Вкладка «Инфо» показывает описание разрабатываемой системы и другую подобную информацию.

1.2.3 Создание кнопки

Следующий шаг – создание кнопок для управления моделью. В поле следует нажать на список объектов интерфейса и выбрать объект «кнопка».

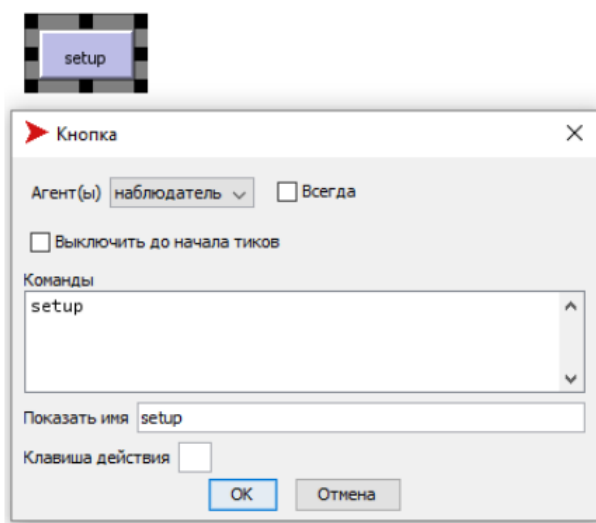


Рис. 1.10: Объекты интерфейса для добавления

Далее создаются рычажки-переключатели, с помощью которых можно изменять различные параметры для агентов и тем самым изменять модель. В списке объектов интерфейса сразу после кнопки находится рычажок. В модели будет расположено 5 таких рычажков. Первый рычажок – количество стеков у сервера. В поле редактирования прописывается название, начальное значение, можно отметить максимум и минимум (рис. 1.11).

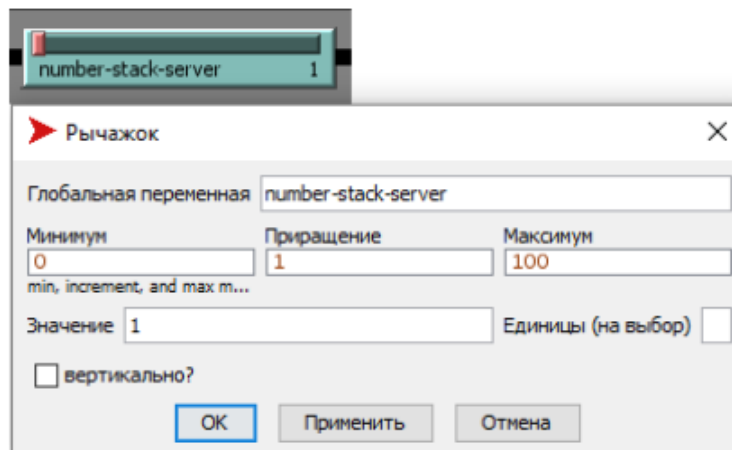


Рис. 1.11: Редактирование рычажка «number-stack-server»

Также создадим рычажки для:

- количество атак (зараженных компьютеров);
- энергии сервера (количество атак, которые способен выдержать сервер) ;
- энергии атаки (модель количества одновременных атак зараженного компьютера);
- воспроизведения новых атак.

Для того чтобы посмотреть поведение сервера и атаки следует добавить график, выбрав его из списка объектов интерфейса. В окне редактирования прописывается название графика, указывается имена осей X, Y и их минимальное и максимальное значения. Также следует добавить 2 пера, которые будут обозначать сервер и атаку, при этом еще выбрать цвет линий и прописать перьям команды, в которых будет прописано, к какому агенту относится перо (рис. 1.12)

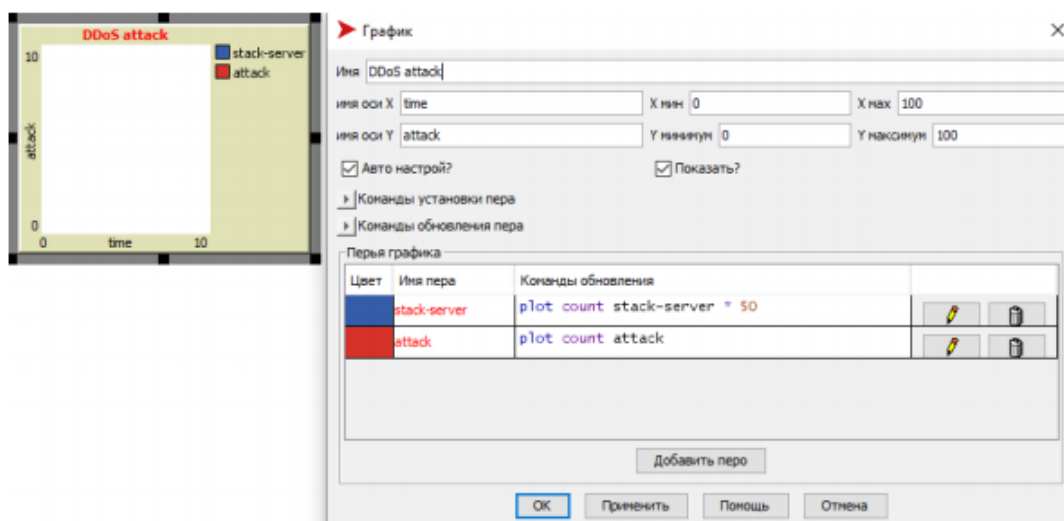


Рис. 1.12: Редактирование графика

Для того чтобы отследить количество атак и количество стеков у сервера следует добавить объект под названием «экран». Он находится в списке объектов интерфейса для добавления. В

окне редактирования прописывается датчик, имя экрана, размер шрифта и количество десятичных знаков. Создается 2 таких объекта, один для сервера, второй для количества атак(рис. 1.13).

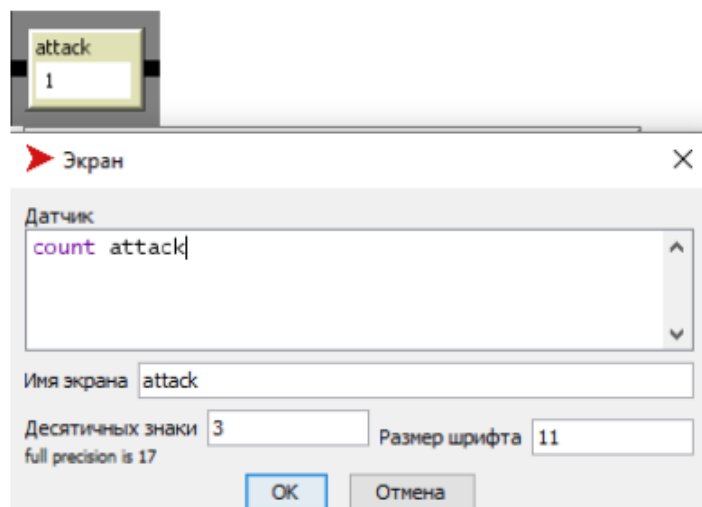


Рис. 1.13: Редактирование объекта для показа количества атак

После добавления объектов интерфейс модели должен выглядеть, как показано на рисунке 1.14.

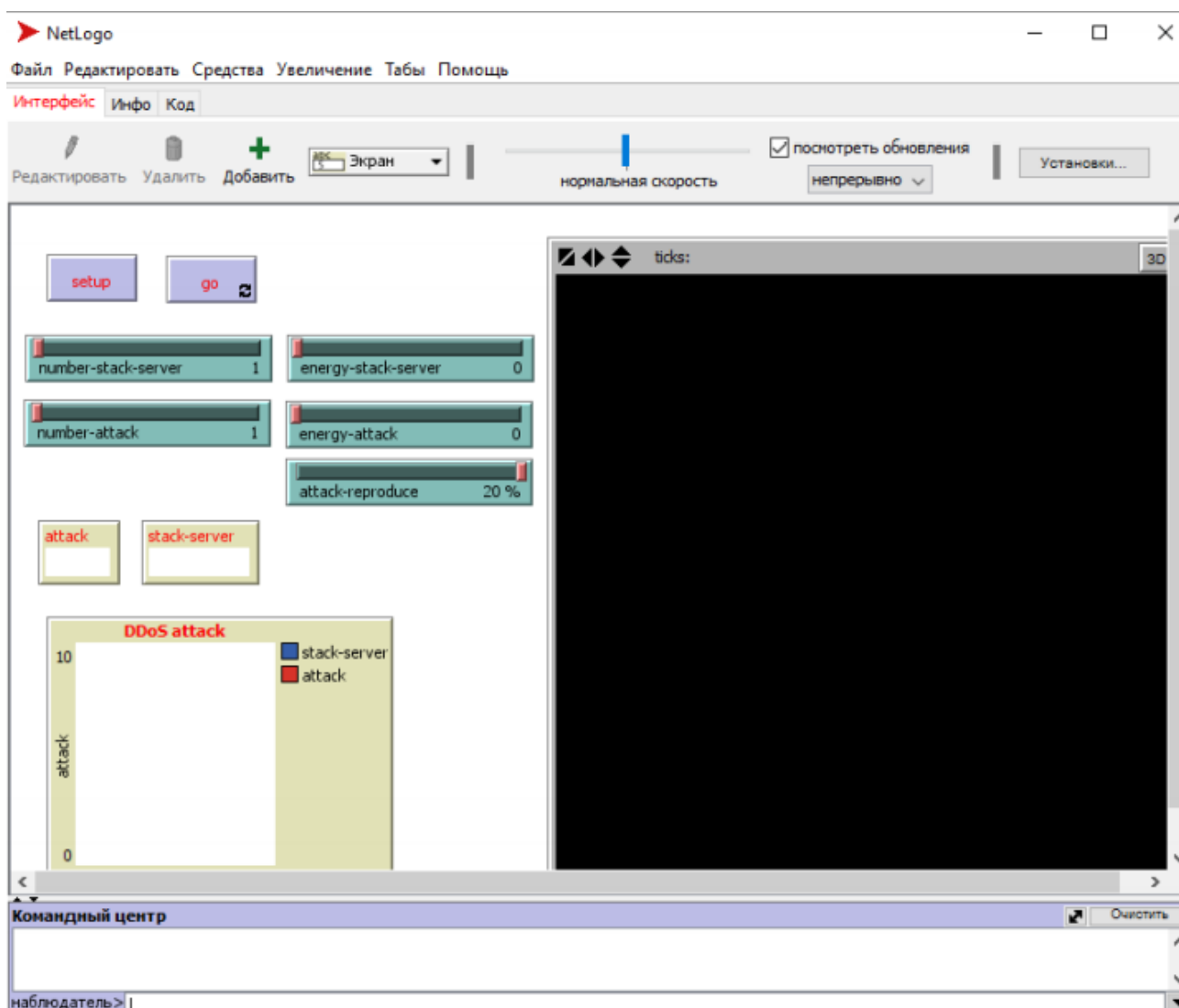


Рис. 1.14: Интерфейс модели после добавления объектов

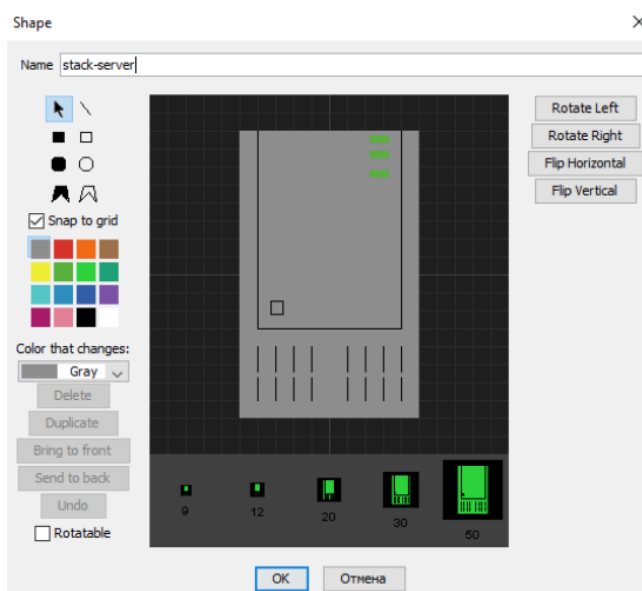
Следующий шаг в создании модели состоит в создании самих агентов.

Для этого в верхней панели нужно выбрать вкладку «Средства» и после Редактор форм черепах». В редакторе форм черепах находятся различные формы, которые можно присвоить агентам, например животные, растения, фигуры и т.д. (рис. 1.15). Но в библиотеке таких форм намного больше, поэтому внизу следует нажать на кнопку «Импорт из библиотеки».

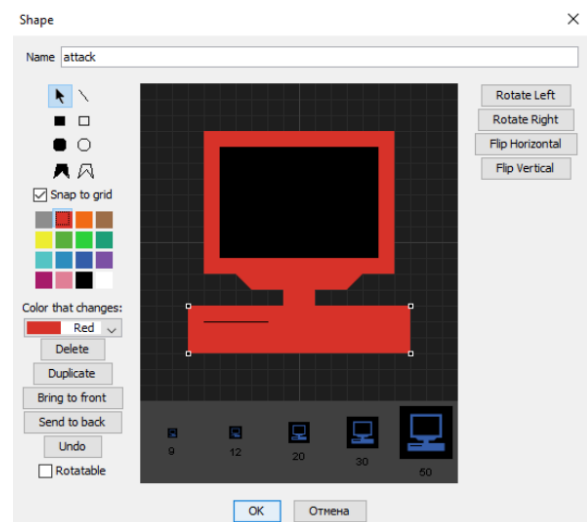


Рис. 1.15: Редактор форм черепах

В библиотеке нужно найти формы «computer server», «computer workstation» и импортировать их в редактор форм. Формы можно дублировать, редактировать, менять цвет, форму, уменьшать и увеличивать, а также менять название. В форме «computer server» изменяется только название на «stack-server» (рис. 1.16a). А у формы «computer workstation» изменяется не только название на «attack», но и меняется цвет на красный, для того, чтобы было понятно, что этот компьютер заражен (рис. 1.16b).



(a) Редактирование формы «stack-server»



(b) Редактирование формы «attack»

Рис. 1.16: Результат работы

После закрытия формы редактора черепах, необходимо перейти к самому коду, где будут

описаны все процедуры, поэтому следует открыть вкладку «код» в верхней панели. Вначале в коде следует описать переменные:

```
1 globals [system max-stack-server]
2 breed [stack-server]
3 breed [attack]
4 turtles-own [energy]
```

Переменная `system max-stack-server` является глобальной. Если переменная глобальная, то она имеет единственное значение, и каждый агент имеет доступ к этому значению.

С помощью ключевого слова «breed» можно определить породу агентов-черепах. Здесь записывается имя переменной такое же, как и при изменении агентов в редактировании форм черепах.

Переменная `energy` относится к агентам и означает, что каждая черепашка обладает собственной энергией.

Для того чтобы модель обновлялась в программном коде следует прописать процедуру «setup», в которой будет содержаться описание параметров сервера и зараженного компьютера, цвет фона, а также форма агентов.

Листинг 1.1: "Процедура «setup»"

```
1 to setup
2   clear-all
3   set max-stack-server 100000
4   ask patches [ set pcolor white ]
5   set-default-shape stack-server "stack-server"
6   create-stack-server number-stack-server
7   [
8     set size 4
9     set energy 1000
10    setxy 0 0
11  ]
12  set-default-shape attack "attack"
13  [
14    set size 2
15    set energy random (2* energy-attack)
16    setxy 0 0
17  ]
18  display-labels
19  set sistem count patches with [pcolor = white]
```

```

20   reset-ticks
21 end

```

С помощью процедуры «go» модель можно воспроизвести. В коде прописываются, какие действия могут выполнять сервер и зараженные компьютеры.

Листинг 1.2: "Процедура «go»"

```

1  to go
2    if not any? turtles [stop]
3    ask stack-server [
4      deash
5    ]
6    ask attack [
7      catch-stack-server
8      death
9      reproduce-attack
10   ]
11   tick
12   display-labels
13 end

```

Для того чтобы зараженных компьютеров становилось больше, следует записать в коде процедуру их размножения, в которой описано, что размножение происходит в случайном порядке и каждый раз когда происходит появление новых атак, то энергия делиться пополам.

Листинг 1.3: "Процедура «reproduce-attack»"

```

1  to reproduce-attack
2    if random-float 100 < attack-reproduce [
3      set energy random (energy-attack/ 2)
4      hatch 1 [ rt 0 fd 0]
5    ]
6  end

```

Следующая процедура отвечает за выведение сервера из строя. Когда зараженные компьютеры атакуют, у сервера уменьшается энергия. В данной процедуре описывается, что если сервер начинают атаковать, то у сервера отнимается энергия на единицу и прибавляется к энергии атаки. Конструкция «let prey one-of stack-server-here» создает переменную prey, которая соотносит себя с любым агентом типа stack-server существующим в рамках вселенной, на которую наткнулся другой агент в данной точке

Листинг 1.4: "Процедура «catch-stack-server»"

```

1  to catch-stack-server
2    let prey one-of stack-server-here
3    ifelse prey != nobody

```

```

4      [ ask prey [
5          set energy energy - 1 ]
6          set energy energy + energy - attack]
7      [set energy energy - energy - attack]
8  end

```

Процедура «death» отвечает за гибель агента. Если энергия сервера будет равна 0, то сервер соответственно выведен из строя и агент исчезает из окна «мира».

В последней процедуре «display-labels» записано обращение к агентам и вывод их количества на объекты «экран» и создание графика в поле интерфейса модели/

Листинг 1.5: "Процедуры «death» и «display-labels»"

```

1  to death
2      if energy < 0 [die]
3  end
4
5  to display-labels
6      ask turtles [ set label "" ]
7  end

```

Запустим модель и увидим рисунок 1.17

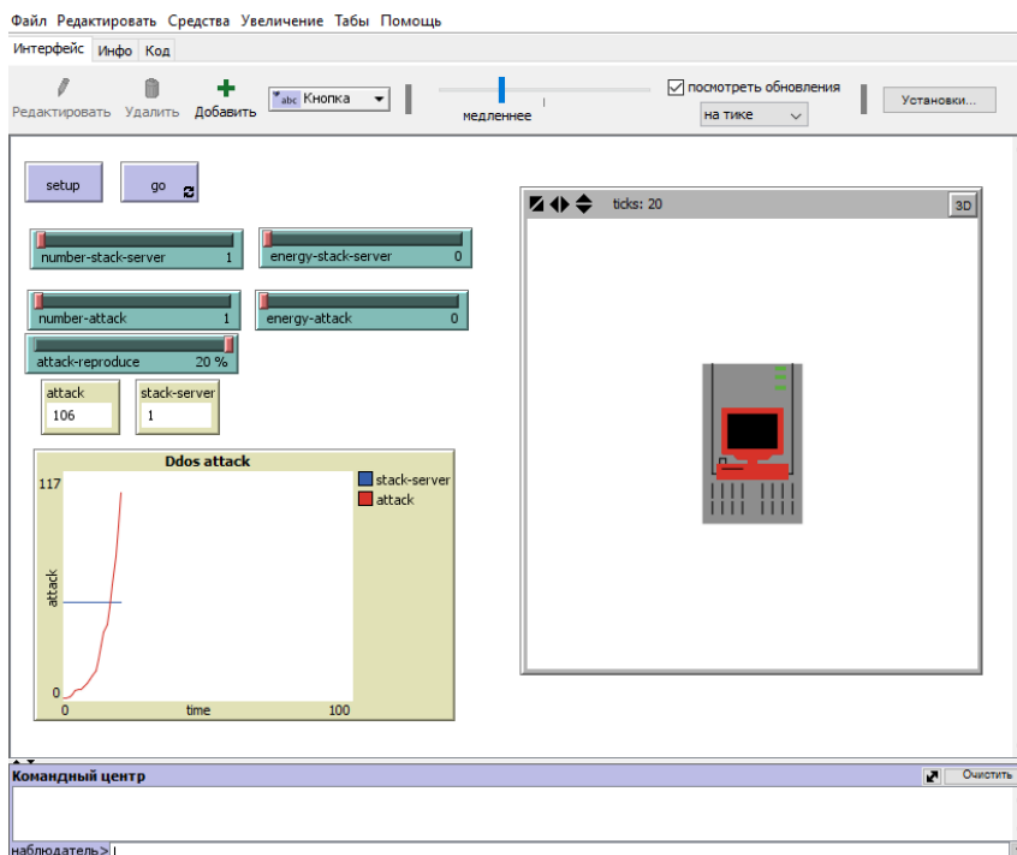


Рис. 1.17: Интерфейс модели DDoS-атаки

После создания модели было проведено исследование, в котором было определено за сколько

тиков атака выведет сервер из строя, если защита для сервера отсутствовала. В таблице(1.18)) ниже представлены результаты исследования без защиты для сервера при разном количестве первоначальных атак

№	Первоначальное количество атак	Количество атак после того как сервер упадет	Количество тиков
1.	1	282	34
2.	15	346	19
3.	25	357	16

Рис. 1.18: Данные исследования

Таким образом, можно сделать вывод, чем выше количество первоначальных атак, тем меньше времени занимает выведение сервера из строя.

1.3 Выводы

В данной работе были изучены многоагентные интеллектуальные системы. В ходе работы было выявлено, что преимуществами МАС являются: гибкость этих систем, способность к самоорганизации и самоуправлению, распределенность данных и децентрализованность системы. В качестве недостатков системы можно указать, что не все системы можно легко представить в модели МАС.

Для разработки МАС можно использовать множество различных инструментов. В том числе, можно использовать как привычные ЯП (C, C++, Java и т.д.), так и различные специализированные платформы (NetLogo, StarLogo и др.).

Мне кажется, что область исследований, связанная с МАС является перспективной, т.к. она позволяет смоделировать и решить большой спектр задач из различных областей науки.

По результатам проведённой работы можно заключить, что моделирование МАС на одной вычислительной машине методом итеративного перебора агентов является корректным, т.к. в данном случае работа системы естественным образом разделяется на «тики» (итерации цикла), поэтому в работе системы не произойдет ошибок, связанных с синхронизацией. Минус такого подхода — низкая скорость работы.

Моделирование МАС путем создания процесса (или потока) для каждого процесса будет корректным, если при этом обеспечивается синхронизация между этими процессами.

Далее в данной работе была проанализирована платформа для разработки и моделирования МАС NetLogo. На основе анализа можно сказать, что NetLogo — достаточно простая и удобная система. Однако, несмотря на свою простоту она хорошо подходит для моделирования сложных систем, развивающихся во времени. Пользователи могут давать

инструкции большому количеству агентов, которые будут выполнять эти инструкции одновременно. Такое моделирование дает возможность изучать связи между поведением отдельных индивидов и результатами взаимодействия индивидов на макроуровне.

Была разработана простая модель DDoS-атаки, которая реализована в мультиагентной среде NetLogo. Кроме этого были проведены исследования, с помощью которых можно увидеть поведение сервера и зараженных компьютеров (атак) и посмотреть за какое время сервер будет выведен из строя.

Список литературы

- [1] *Многоагентные системы*. URL: <http://www.aiportal.ru/articles/multiagent-systems/multiagent-systems.html> (дата обр. 29.11.2017).
- [2] *Официальный сайт и документация NetLogo*. URL: <http://ccl.northwestern.edu/netlogo/> (дата обр. 29.11.2017).
- [3] *Применение технологии многоагентных систем для интеллектуальной поддержки принятия решения*. URL: <http://systech.miem.edu.ru/2003/n1/Chekinov.htm> (дата обр. 29.11.2017).
- [4] В. Ф. Хорошевский Т. А. Гаврилова. *Базы знаний интеллектуальных систем*. СПб: Питер, 2000.