

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА** - **Российский технологический университет"**

**РТУ МИРЭА**



**Институт** Информационных Технологий

**Кафедра** Вычислительной Техники

**Практические работы по дисциплине**

**«Многоагентное моделирование»**

Студент группы: ИКБО-04-22 \_\_Трофимов А.А.\_\_

*(Фамилия студента)*

Преподаватель \_\_Гололобов А.А.\_\_\_

*(Фамилия преподавателя)*

Москва 2024

**ВВЕДЕНИЕ**

Современная открытая экономика и информационное общество ускоряют научно-технический прогресс и усиливают конкуренцию, вынуждая компании искать более гибкие методы управления. Многие предприятия сталкиваются с проблемой оперативного выявления новых потребностей и адаптации ресурсов (производственных, кадровых, финансовых) к изменяющимся условиям. Такая необходимость возникает, например, при получении выгодного заказа, нехватке мощностей, поломках оборудования или изменении критериев принятия решений.

Чем выше неопределенность и распределенность процессов, тем менее эффективны традиционные системы, неспособные к автономной адаптации. Их модификация требует значительных затрат и высокой квалификации специалистов.

Решением являются мультиагентные технологии, основанные на концепции автономных агентов. Эти подходы находят применение в программировании, искусственном интеллекте и распределенных системах, позволяя создавать более гибкие и адаптивные управленческие решения.

**1 МНОГОАГЕНТНОЕ МОДЕЛИРОВАНИЕ**

**1.1 Постановка задачи**

Необходимо построить модель, описывающую распространение различных СМИ. С точки зрения реализации, каждый отдельный человек является агентом. Изначально все агенты являются потенциальными пользователями. СМИ распространяется с заданной эффективностью и могут также распространяться между людьми, также пользователи иногда могут сами сменить СМИ. Можно настроить уровень цифровизации населения и ввести санкции.

Требования к модели:

* Количество состояний агента – не менее 3;
* Количество параметров агента - не менее 5;
* Размер популяции агентов – не менее 500;
* Возможность динамического изменения параметров – не менее 1;
* Наличие условий в карте агента: не менее 1;
* Наличие графика/ов, для отслеживания динамики изменения состояний популяции.

**1.2 Описание этапов выполнения работы**

Была создана популяция агентов и выбраны три вида СМИ(TV, Telegram, X), затем была построена диаграмма состояний, задающая поведение агентам (Рисунок 1.2.1).

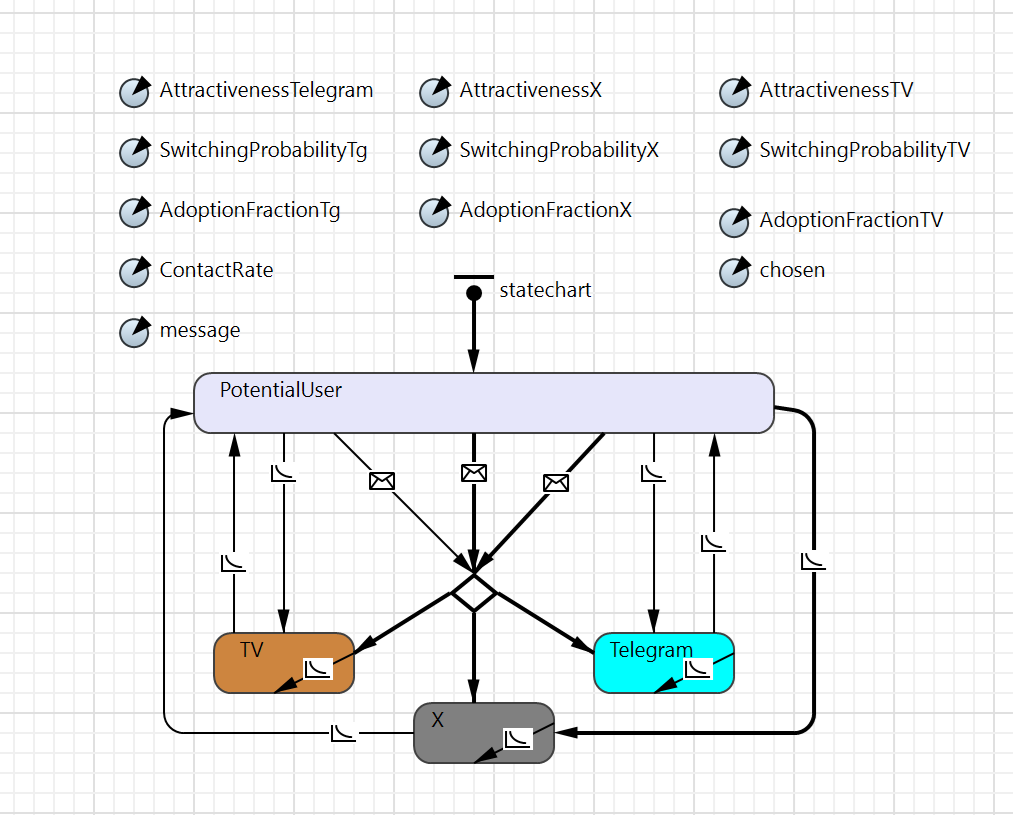


Рисунок 1.2.1 – Диаграмма состояний

Потенциальный пользователь выбирает СМИ, согласно интенсивности распространения (Рисунок 1.2.2)

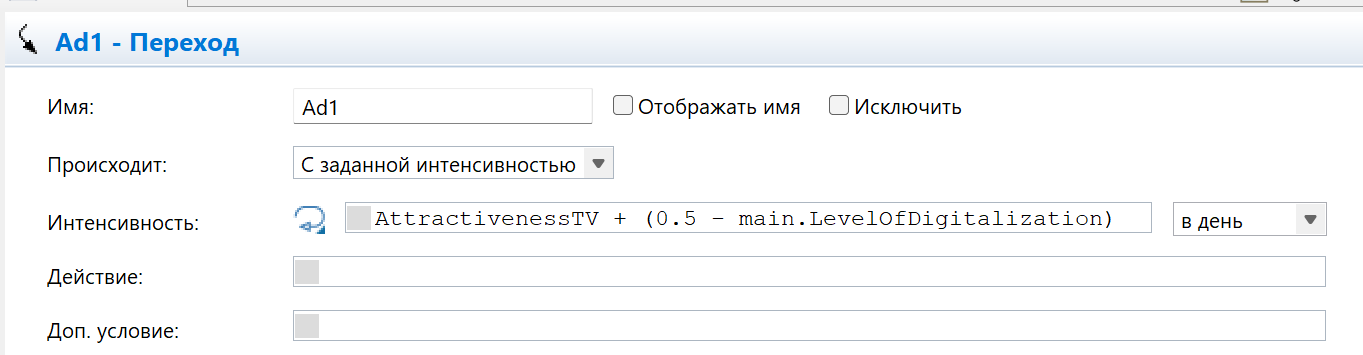


Рисунок 1.2.2 – переход Ad1

Также, потенциальный человек может распространить СМИ на другого человека (Рисунок 1.2.3)

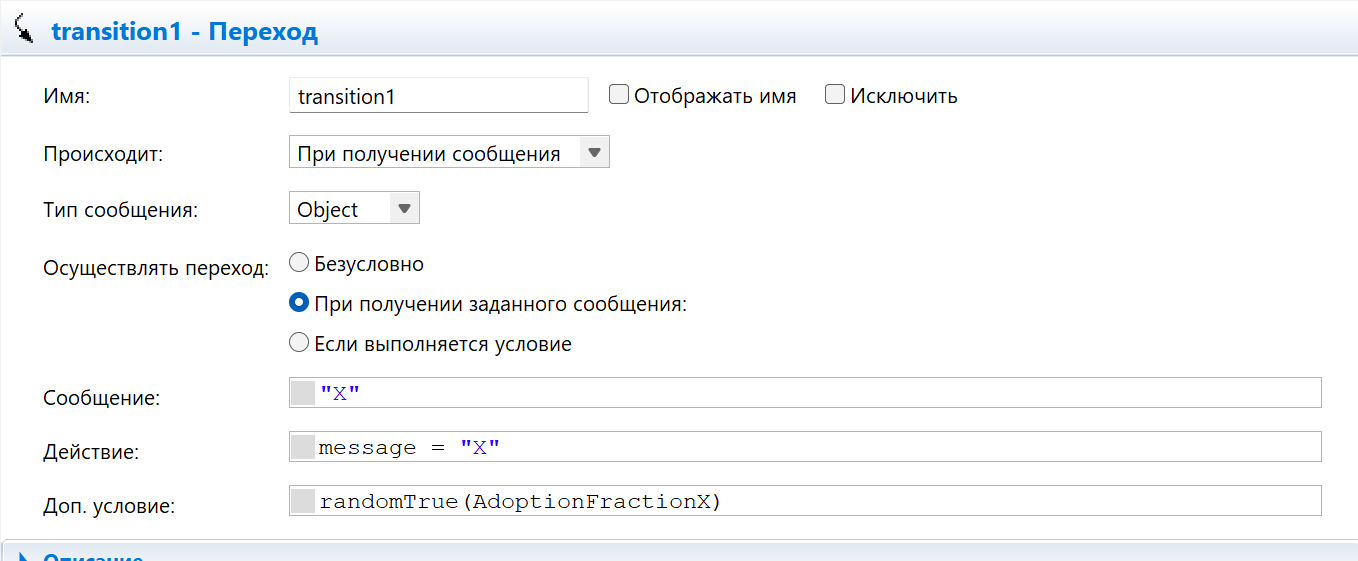


Рисунок 1.2.3 – переход transition1

Через некоторое время пользователи могут менять предпочтения в СМИ (Рисунок 1.2.4)

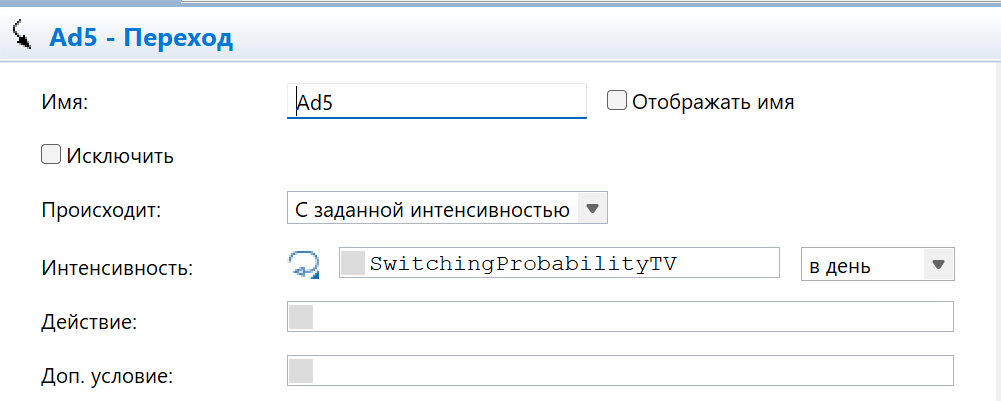


Рисунок 1.2.4 – переход Ad5

Для изменения модели добавлены ползунок уровень цифровизации населения, увеличивающий популярность TV при уменьшении, и кнопку “Санкции”, уменьшающую скорость распространения X.(Рисунок 1.2.5)

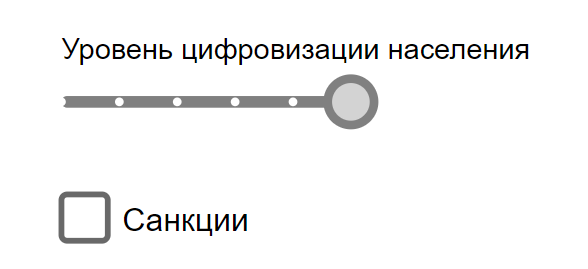


Рисунок 1.2.5 – ползунок и кнопка

Для визуализации динамики системы была добавлена временная диаграмма с накоплением (Рисунок 1.2.6).

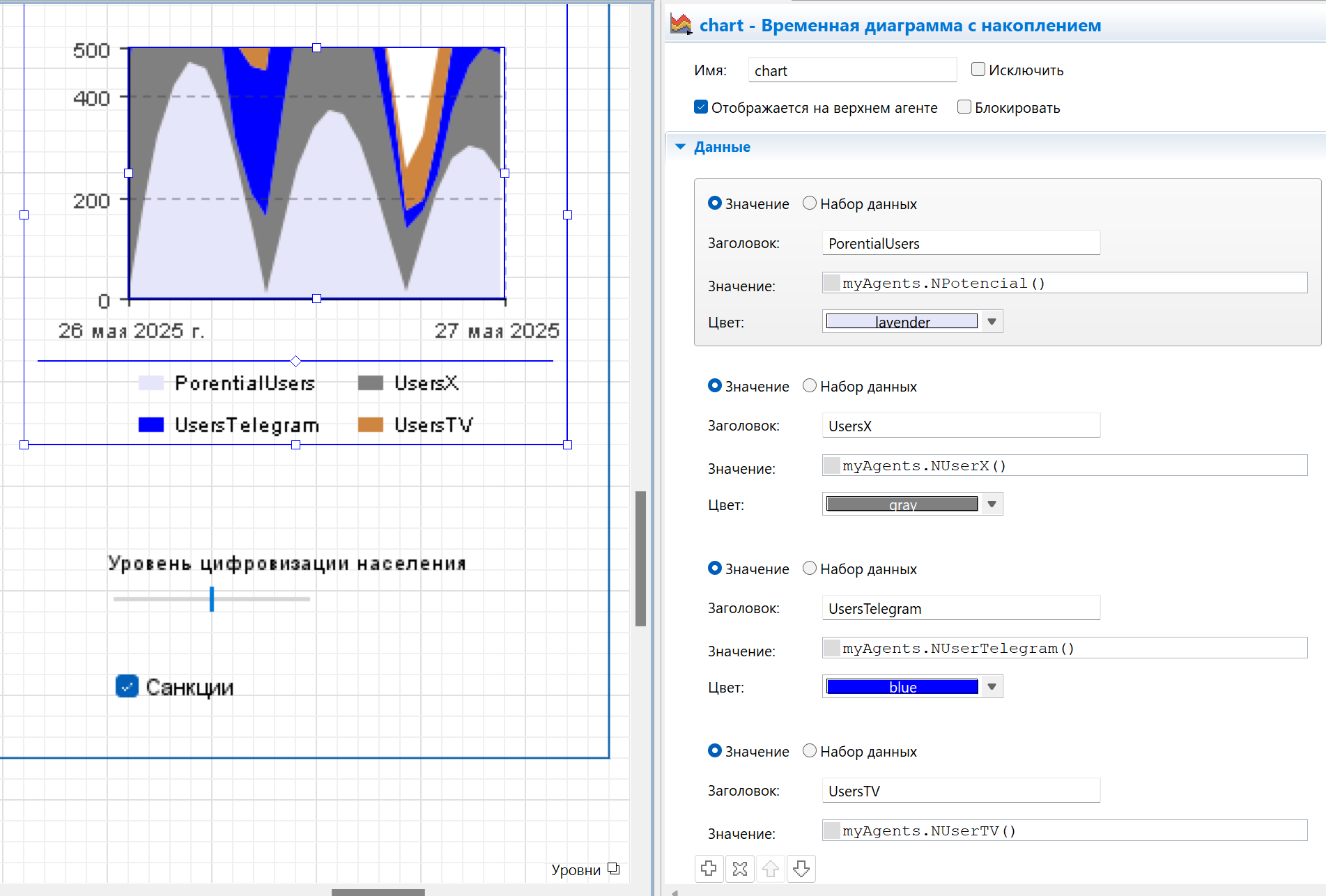


Рисунок 1.2.6 – Временная диаграмма с накоплением

Работа модели представлена на Рисунке 1.2.7

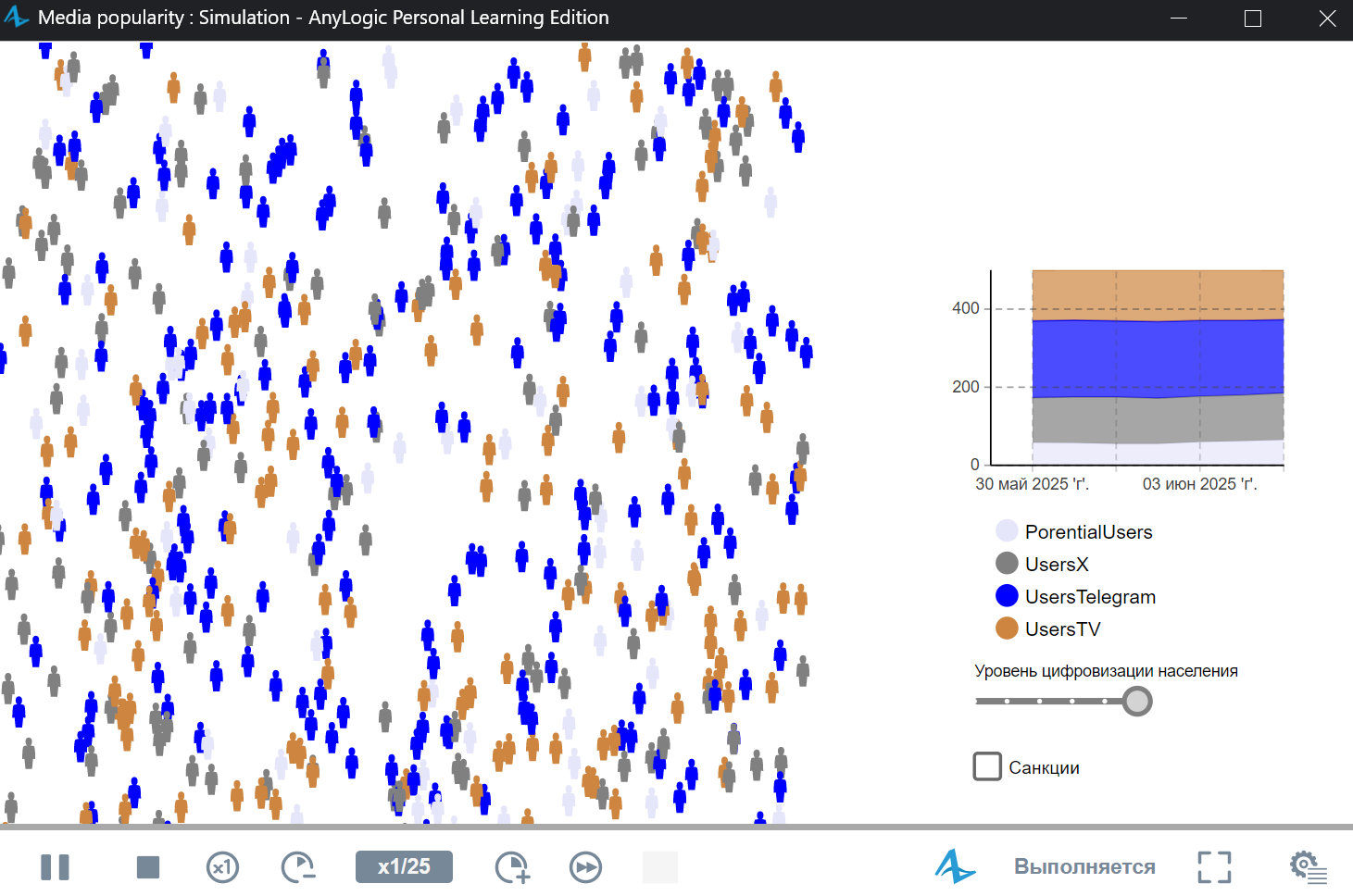


Рисунок 1.2.7 – работа модели

**2 ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ**

**2.1 Постановка задачи**

Необходимо промоделировать работу Ж/Д вокзала. Учесть возможность покупки билетов в автоматах. Добавить металлоискатели на входе. Далее пассажиры будут ожидать свой поезд и зайдут в него когда он приедет.

Требования к модели:

* Количество состояний агента – не имеет значения;
* Количество параметров агента – не имеет значения;
* Размер популяции агентов – не менее 10;
* Возможность динамического изменения параметров – не менее 1;
* Наличие условий в карте агента: не менее 1;
* Наличие картинки и 3D окна, для отслеживания динамики изменения состояний популяции.

**2.2 Описание этапов выполнения работы**

Изображение модели представлено на Рисунке 2.2.1.

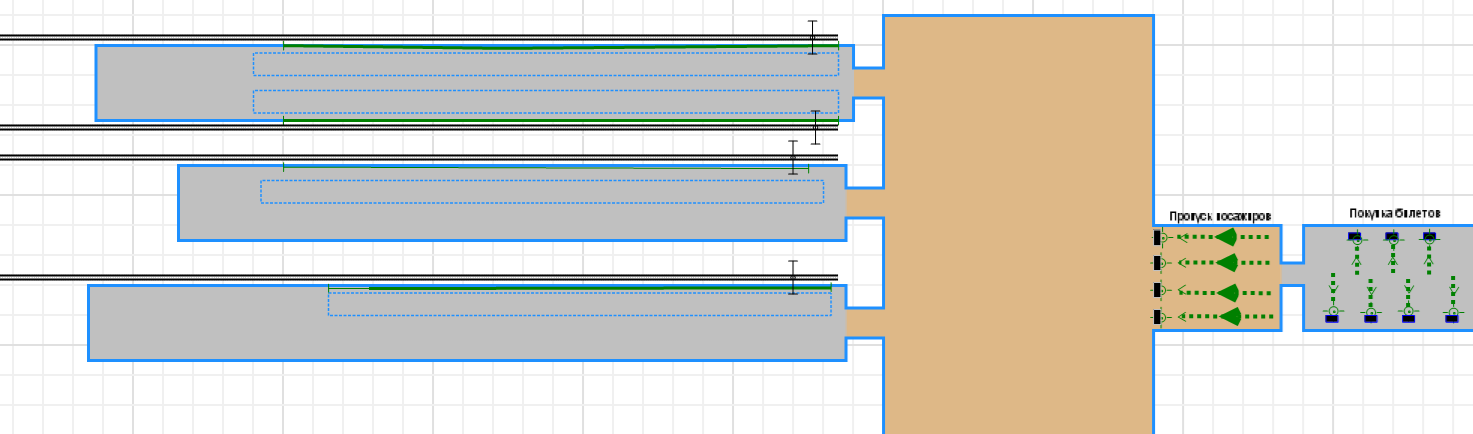
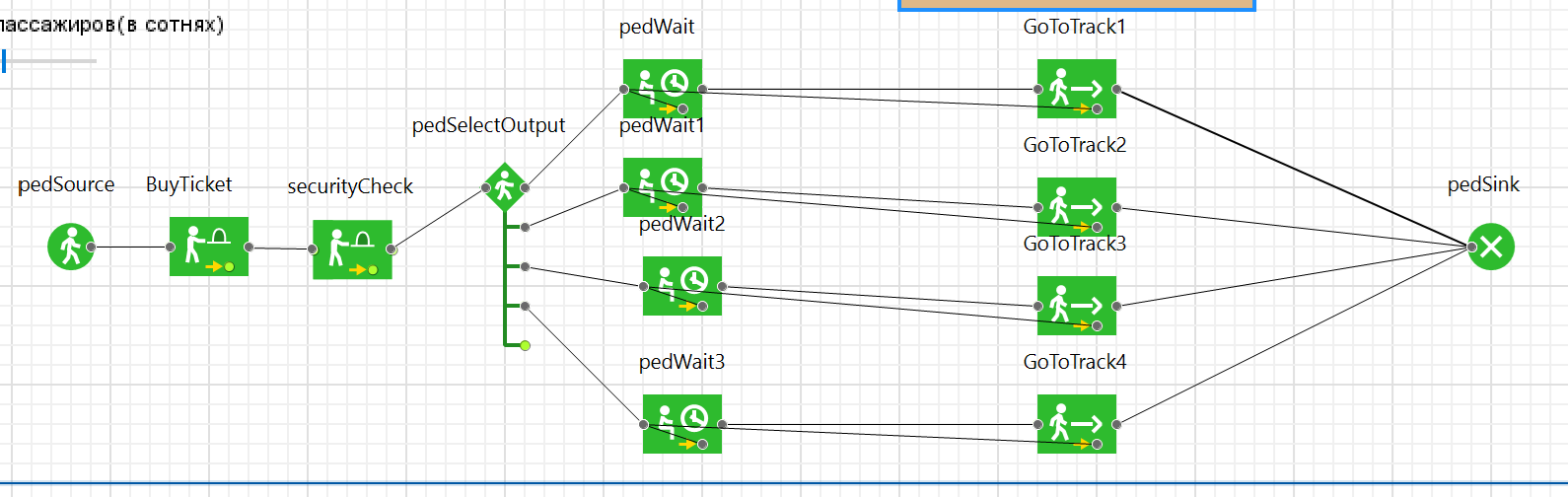


Рисунок 2.2.1 – изображение модели

Для диаграммы переходов использовалась пешеходная библиотека, сначала агенты появляются с помощью PedSource, потом проходят несколько этапов и, когда приезжает поезд заходят в него. Диаграмма переходов представлена на Рисунке 2.2.2.



Для моделирования работы поездов использовалась Железнодорожная библиотека, поезд появляется на точке жд пути после ожидания приезжает на другую точку, во время пребывания на ней в поезд заходят пассажиры, а потом он уезжает обратно.

Система работы поездов представлена на Рисунке 2.2.3



Рисунок 2.2.3 - Система работы поездов

С помощью бегунка можно увеличить количество пассажиров в сотнях в час, от 1 до 4, за это отвечает параметр NumberOfPassengers (Рисунок 2.2.4)

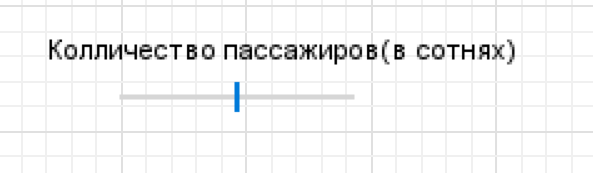


Рисунок 2.2.4 – количество пассажиров

Запуск модели представлен на рисунке 2.2.5

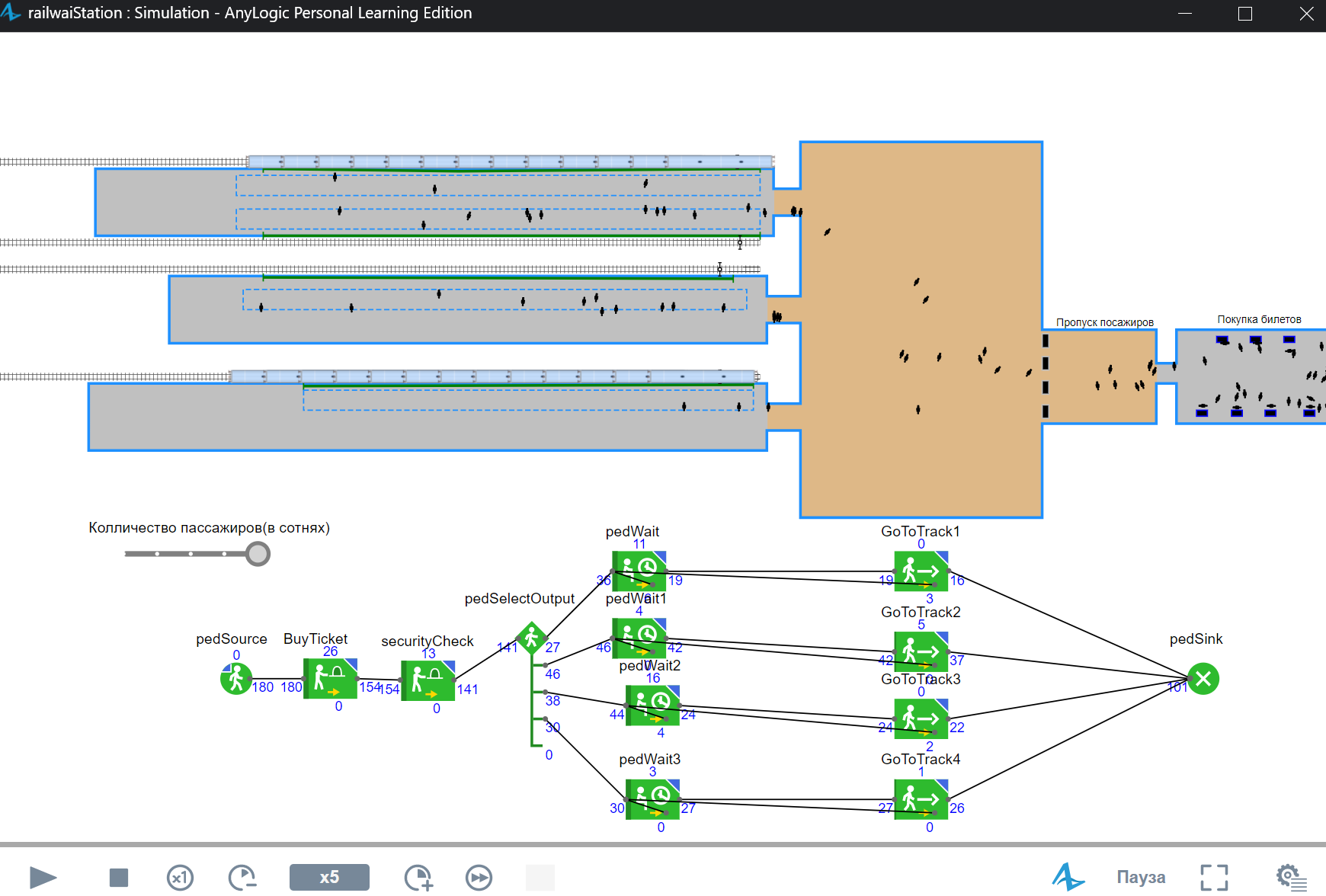


Рисунок 2.2.5 – запуск модели

**3 СИСТЕМНАЯ ДИНАМИКА**

**3.1 Постановка задачи**

Разработать модель, которая описывает систему управления энергетикой, учитывать взаимодействие между производящими энергию зданиями и зданиями, которым нужна энергия.

Требования к модели:

* Количество накопителей – не менее 4;
* Количество потоков - не менее 3;
* Количество параметров агента – не менее 1;
* Возможность динамического изменения параметров – не менее 1;
* Наличие обратных связей – потоков (циклов);
* Наличие графика/ов, для отслеживания динамики изменения состояний популяции.

**3.2 Описание этапов выполнения работы**

Для создания энергии в модели созданы 4 предприятия для производства энергии (ТЭС, ГЭС, Ветряные турбины и АЭС), ТЭС тратит газ для производства энергии, АЭС тратит ядерное топливо, а ГЭС и Ветряные турбины ничего не используют просто производят энергию.(Русунок 3.2.1)

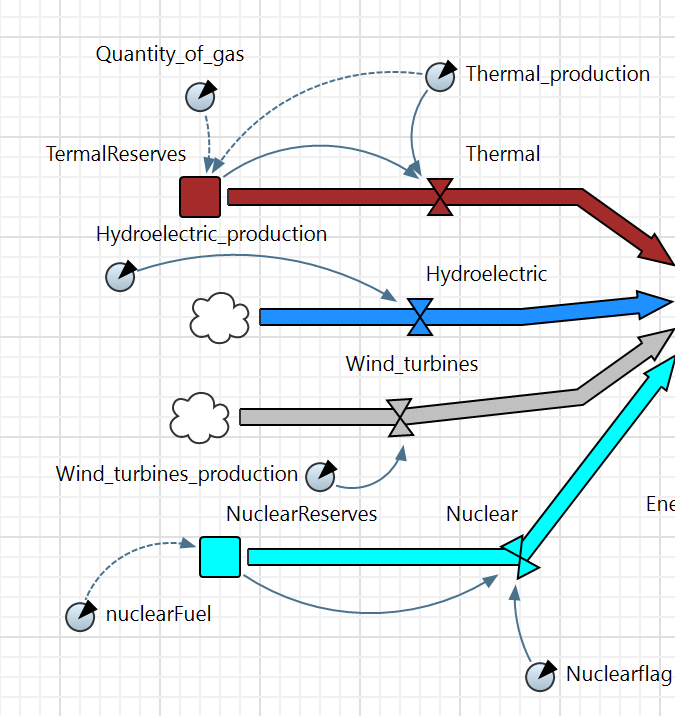


Рисунок 3.2.1 – часть модели для производства энергии

Энергия набирается в EnergyReservs и потом распределяется между школой, больницей, заводом и жилым домом, каждому зданию нужно разное количество энергии, если энергии не хватает на все здания, то школа и завод перестают потреблять энергию, больница и жилой дом в приоритете. (Рисунок 3.2.2)

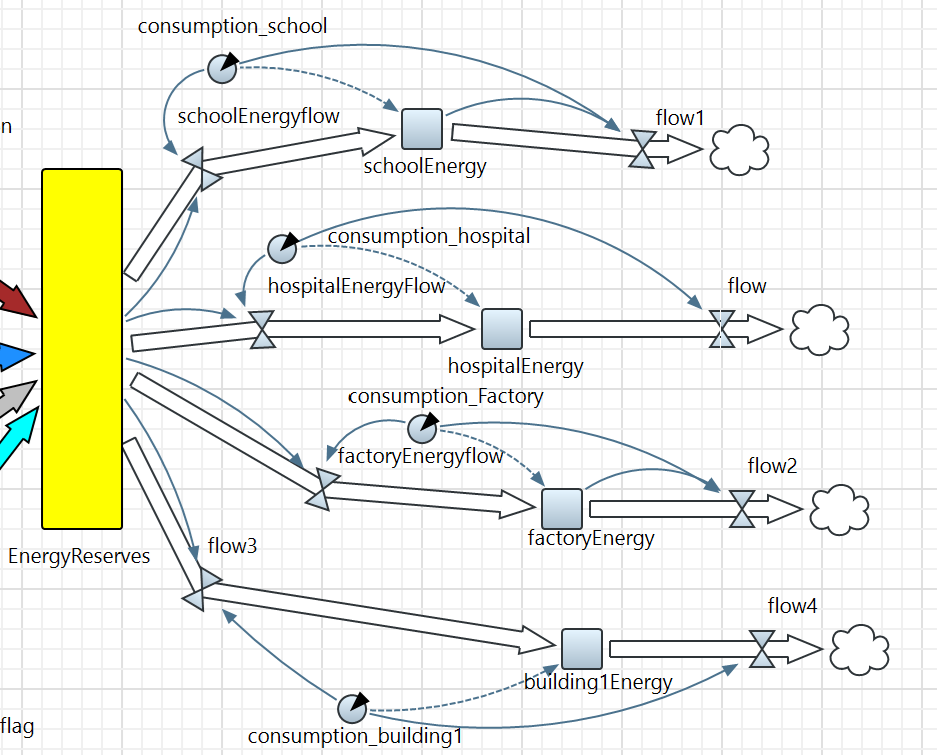


Рисунок 3.2.2 – Запас энергии и его распределение

Также были добавлены график, регуляторы производства ГЭС и турбин, кнопка для включения и выключения АЭС, и сброса запасов газа и ядерного топлива(Рисунок 3.2.3)

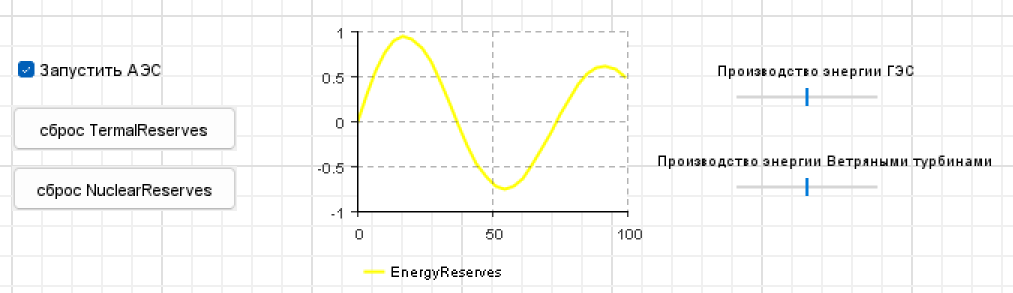


Рисунок 3.2.3 – график и элементы для изменения модели

Запуск модели системной динамики представлен на Рисунке 3.2.4.

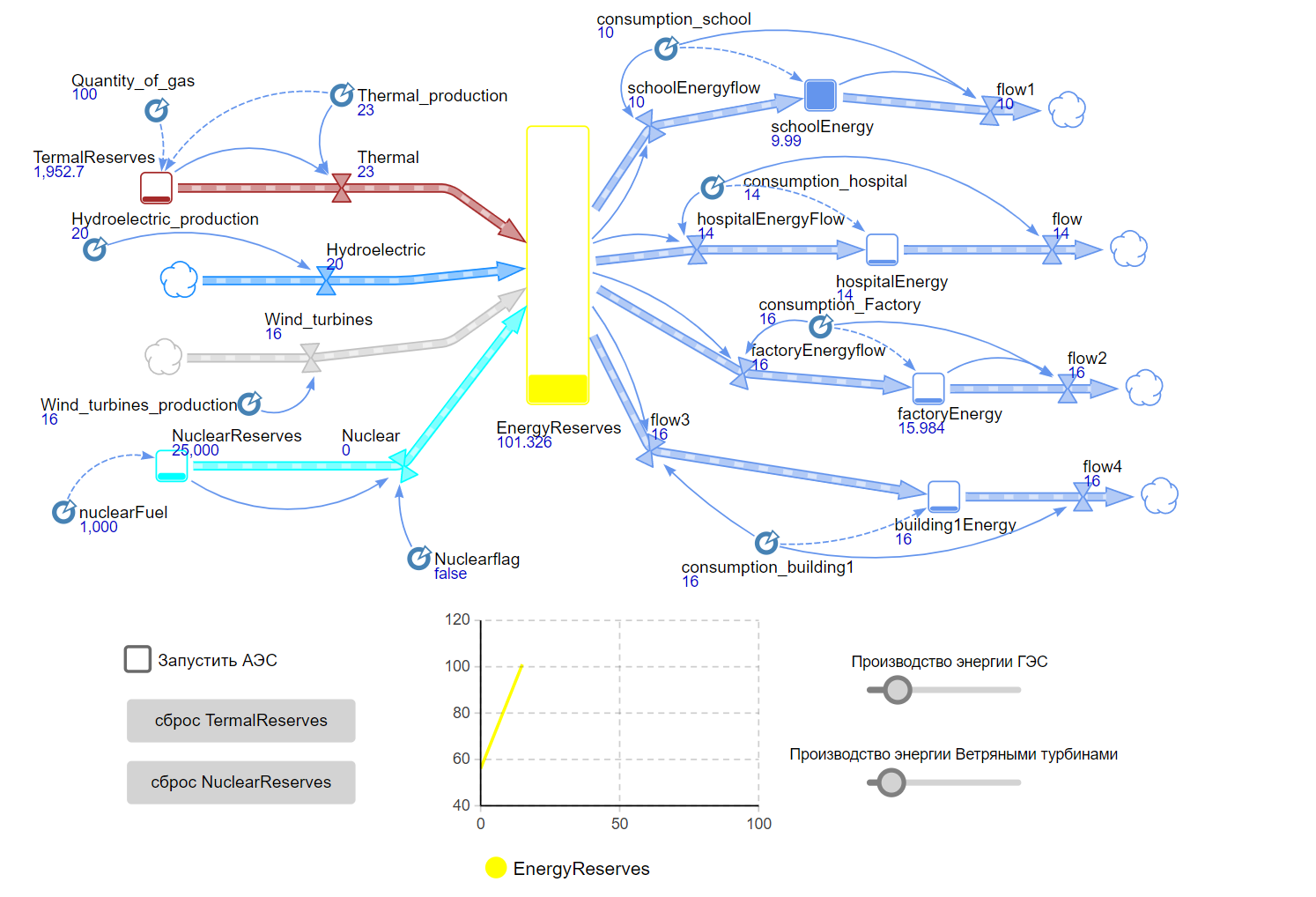


Рисунок 3.2.4 – запуск модели

На Рисунке 3.2.5 представлено как она работает при недостатке производства энергии, школа и завод перестают работать

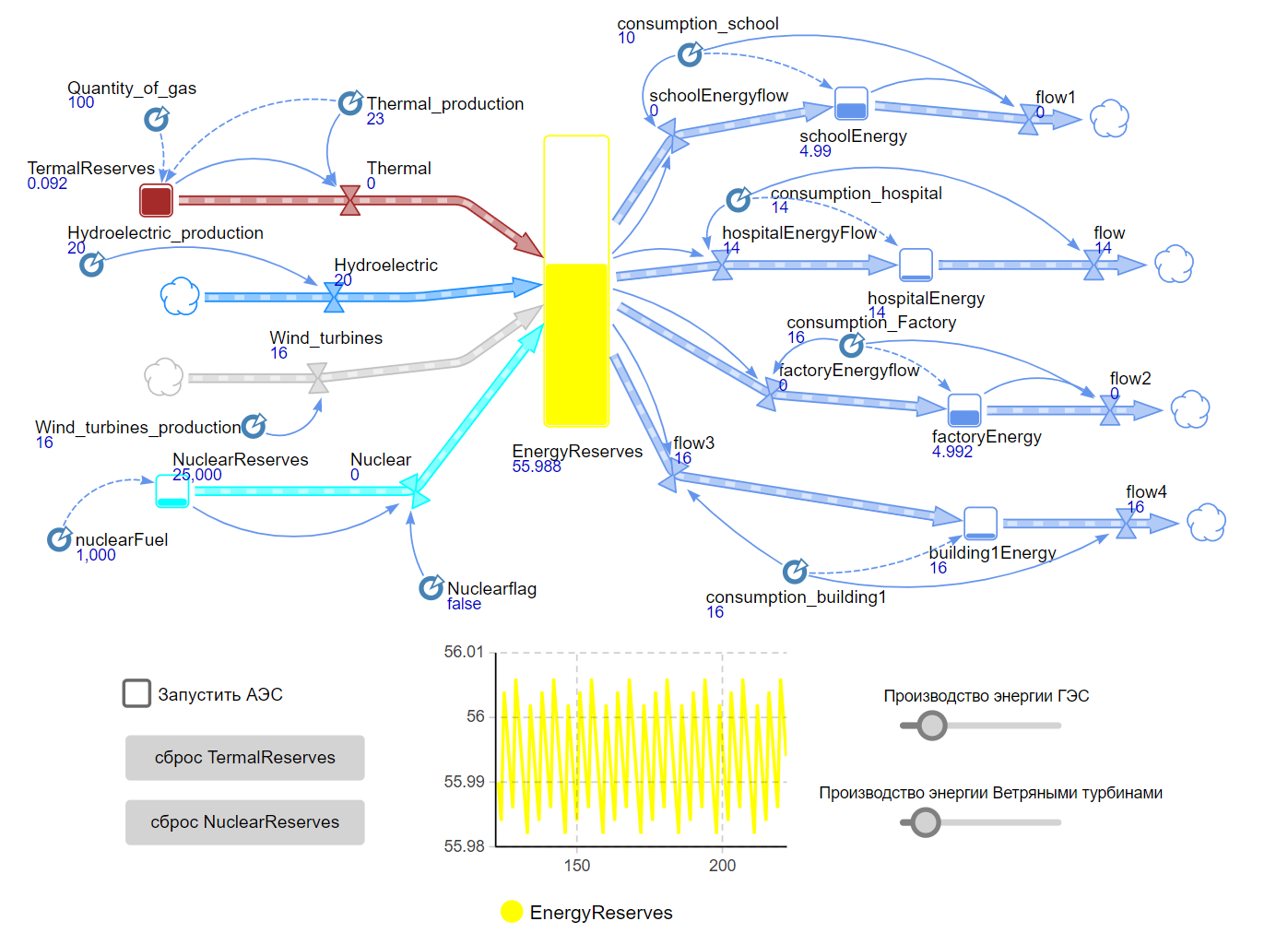


Рисунок 3.2.5 – работа модели при дефиците производства энергии

**4 ЗАДАЧА НА ПРОГРАММИРОВАНИЕ**

**4.1 Постановка задачи**

Необходимо построить модель, описывающую распространение различных СМИ. С точки зрения реализации, каждый отдельный человек является агентом. Изначально все агенты являются потенциальными пользователями. СМИ распространяется с заданной эффективностью и могут также распространяться между людьми, также пользователи иногда могут сами сменить СМИ. Можно настроить уровень цифровизации населения и ввести санкции.

**4.2 Описание этапов выполнения работы**

Для моделирования распространения СМИ была разработана агент-ориентированная модель на Python. Визуально она состоит из двух частей: слева отображаются агенты (люди) с их текущими подписками ("none", "Telegram", "TV" или "X"), а справа — график динамики подписчиков.

Агенты имеют фиксированные случайные координаты и могут менять подписки: неподписанные — с базовой вероятностью, подписанные — переключаться на другое СМИ или рекомендовать своё.

На модель влияют два внешних фактора: уровень цифровизации (увеличивает спрос на цифровые СМИ) и санкции (снижают популярность "X" в 3 раза). Управление осуществляется через ползунок (цифровизация) и кнопку (санкции).

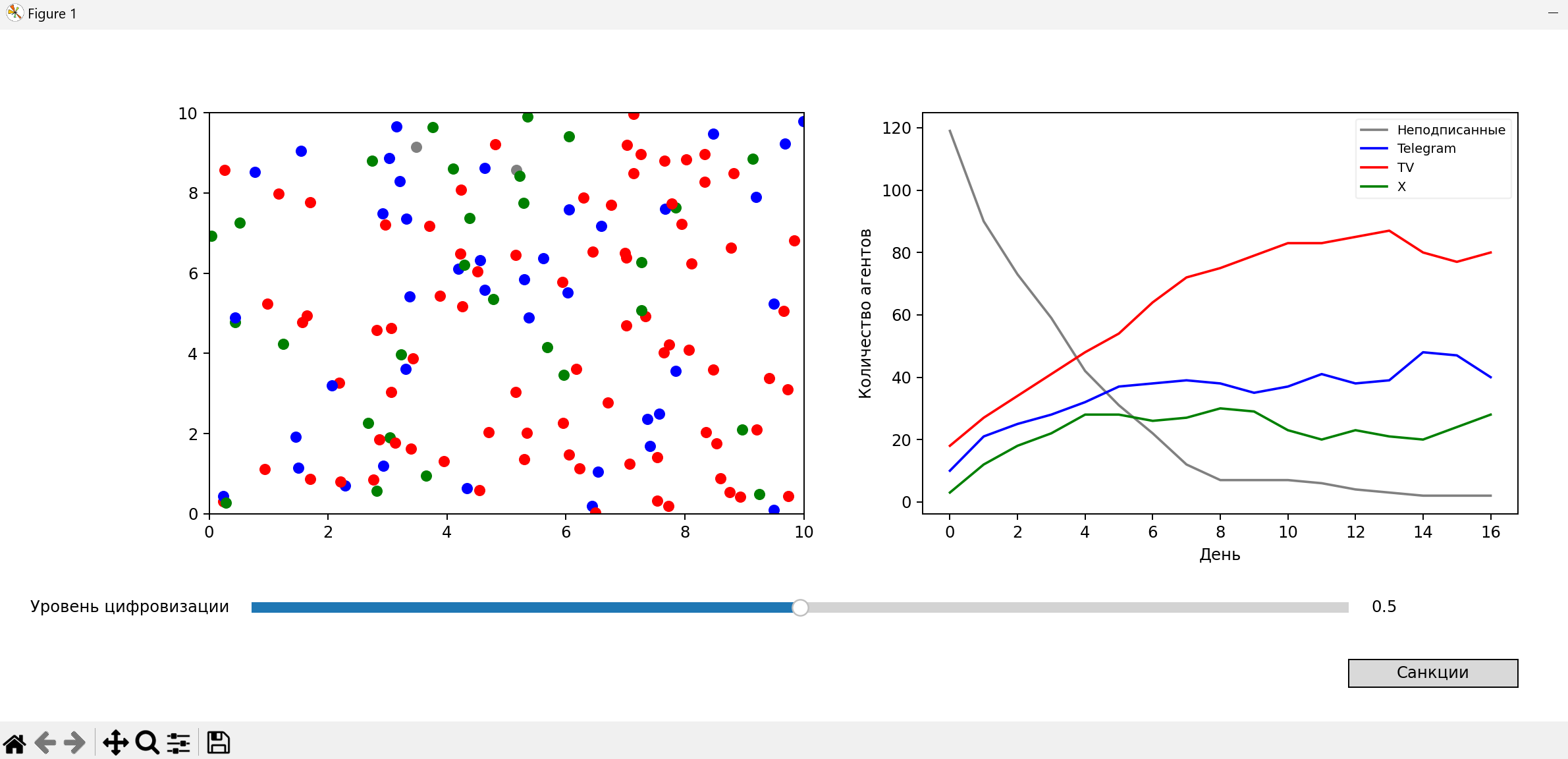


Рисунок 4.2.1 – работа программы

**ВЫВОД**

**Мультиагентные системы** играют ключевую роль в гибком проектировании и управлении производственными, логистическими и e-commerce-системами, а также в обработке знаний. Их важная особенность — социальный аспект взаимодействия агентов, что делает их незаменимыми для решения сложных задач. Такие системы должны работать непрерывно, адаптируясь к новым данным и извлекая знания для изменения своего поведения.

В ходе работы я освоил **AnyLogic** — ведущий инструмент для бизнес-моделирования, поддерживающий три основных подхода:

**Агентное моделирование** — анализ взаимодействия индивидов в системе и их взаимного влияния.

**Дискретно-событийное моделирование** — подходит для процессов с потоками объектов (клиенты, транспорт, пациенты) и применяется в логистике, производстве и здравоохранении.

**Системная динамика** — стратегическое моделирование сложных систем без детализации, полезное для долгосрочного планирования.

Каждый из этих методов эффективен в своей области, позволяя решать задачи разного уровня сложности.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Хливнеко Л.В., Пятакович Ф.А., Практика нейросетевого моделирования, Санкт-Петербург:Лань, 2019, -200с.
2. Рыжков Ю.И. Имитационное Моделирование. Авторская имитация систем и сетей с очередями, Санкт-Петербург:Лань, 2019, -112с.
3. Мезенцев К.Н., Мультиагентное моделирование в среде NetLogo, Санкт-Петербург:Лань, 2015, -176с.
4. Тихвинский В.И., Сорокин А.Б. Имитационное моделирование: практикум [Электронный ресурс]: Учебно-методическое пособие / Тихвинский В.И., Сорокин А.Б. – М.: Московский технологический университет (МИРЭА), 2018, –157с.
5. Палей А.Г., Поллак Г.А., Имитационное моделирование. Разработка имитационных моделей средствами iWebsim и AnyLogic, СанктПетербург:Лань, 2019, – 122с.

**ПРИЛОЖЕНИЯ**

Приложение А — Исходный код практической работы на

программирование

Приложение А

*Листинг А – Код программной реализации задачи*

|  |
| --- |
| import random  import matplotlib.pyplot as plt  import matplotlib.animation as animation  from matplotlib.widgets import Slider, Button  # Параметры симуляции  NUM\_AGENTS = 150 # Количество агентов  BASE\_PROB\_SUBSCRIBE = {  "Telegram": 0.06, # Базовая вероятность подписки на Telegram  "TV": 0.04, # Базовая вероятность подписки на TV  "X": 0.05 # Базовая вероятность подписки на X  }  BASE\_PROB\_SWITCH = {  "Telegram": 0.02, # Базовая вероятность переключения на Telegram  "TV": 0.015, # Базовая вероятность переключения на TV  "X": 0.018 # Базовая вероятность переключения на X  }  PROB\_RECOMMEND = 0.12 # Базовая вероятность рекомендации СМИ  SANCTIONS\_FACTOR = 0.333 # Множитель для вероятностей X при санкциях (1/3)  OBSERVATION\_PERIOD = 180 # Срок наблюдения в днях  # Размеры прямоугольной области  WIDTH = 10  HEIGHT = 10  # День наблюдения  day\_count = 0  # Инициализация уровня цифровизации и состояния санкций  digitalization\_level = 0.5  sanctions\_active = False  # Создание популяции агентов  agents = []  for i in range(NUM\_AGENTS):  agent = {  "media": "none", # Тип СМИ: "none", "Telegram", "TV" или "X"  "x": random.random() \* WIDTH, # Фиксированная x-координата  "y": random.random() \* HEIGHT # Фиксированная y-координата  }  agents.append(agent)  # Инициализация данных для графика  num\_none = [] # Количество неподписанных агентов  num\_telegram = [] # Количество подписчиков Telegram  num\_tv = [] # Количество подписчиков TV  num\_x = [] # Количество подписчиков X  # Функция для пересчета вероятностей на основе уровня цифровизации и санкций  def update\_probabilities(digitalization, sanctions):  prob\_subscribe = {}  prob\_switch = {}    # Модификация вероятностей подписки  tv\_boost = (1 - digitalization) \* 0.1  digital\_boost = digitalization \* 0.1    prob\_subscribe["TV"] = min(BASE\_PROB\_SUBSCRIBE["TV"] + tv\_boost, 0.15)  prob\_subscribe["Telegram"] = max(BASE\_PROB\_SUBSCRIBE["Telegram"] - tv\_boost / 2 + digital\_boost / 2, 0.01)  prob\_subscribe["X"] = max(BASE\_PROB\_SUBSCRIBE["X"] - tv\_boost / 2 + digital\_boost / 2, 0.01)    # Модификация вероятностей переключения  prob\_switch["TV"] = min(BASE\_PROB\_SWITCH["TV"] + tv\_boost / 2, 0.1)  prob\_switch["Telegram"] = max(BASE\_PROB\_SWITCH["Telegram"] - tv\_boost / 4 + digital\_boost / 4, 0.005)  prob\_switch["X"] = max(BASE\_PROB\_SWITCH["X"] - tv\_boost / 4 + digital\_boost / 4, 0.005)    # Применение санкций к X  prob\_recommend\_x = PROB\_RECOMMEND  if sanctions:  prob\_subscribe["X"] \*= SANCTIONS\_FACTOR  prob\_switch["X"] \*= SANCTIONS\_FACTOR  prob\_recommend\_x \*= SANCTIONS\_FACTOR    return prob\_subscribe, prob\_switch, prob\_recommend\_x  # Инициализация вероятностей  prob\_subscribe, prob\_switch, prob\_recommend\_x = update\_probabilities(digitalization\_level, sanctions\_active)  # Функция обновления для анимации  def update(frame):  global day\_count, prob\_subscribe, prob\_switch, prob\_recommend\_x    # Пересчет вероятностей  prob\_subscribe, prob\_switch, prob\_recommend\_x = update\_probabilities(digitalization\_level, sanctions\_active)  # Обновление агентов  for agent in agents:  if agent["media"] == "none":  # Неподписанный агент может подписаться  for media, prob in prob\_subscribe.items():  if random.random() < prob:  agent["media"] = media  break  else:  # Подписанный агент может переключиться  for media, prob in prob\_switch.items():  if media != agent["media"] and random.random() < prob:  agent["media"] = media  break  # Подписанный агент может порекомендовать свое СМИ  recommend\_prob = prob\_recommend\_x if agent["media"] == "X" else PROB\_RECOMMEND  if random.random() < recommend\_prob:  other\_agent = random.choice(agents)  if other\_agent["media"] != agent["media"]:  other\_agent["media"] = agent["media"]  # Подсчет количества агентов по типам СМИ  num\_none.append(len([agent for agent in agents if agent["media"] == "none"]))  num\_telegram.append(len([agent for agent in agents if agent["media"] == "Telegram"]))  num\_tv.append(len([agent for agent in agents if agent["media"] == "TV"]))  num\_x.append(len([agent for agent in agents if agent["media"] == "X"]))  # Отображение агентов  ax1.cla()  ax1.set\_xlim(0, WIDTH)  ax1.set\_ylim(0, HEIGHT)  for agent in agents:  if agent["media"] == "none":  ax1.scatter(agent["x"], agent["y"], c="gray", label="None" if "None" not in ax1.get\_legend\_handles\_labels()[1] else "")  elif agent["media"] == "Telegram":  ax1.scatter(agent["x"], agent["y"], c="blue", label="Telegram" if "Telegram" not in ax1.get\_legend\_handles\_labels()[1] else "")  elif agent["media"] == "TV":  ax1.scatter(agent["x"], agent["y"], c="red", label="TV" if "TV" not in ax1.get\_legend\_handles\_labels()[1] else "")  elif agent["media"] == "X":  ax1.scatter(agent["x"], agent["y"], c="green", label="X" if "X" not in ax1.get\_legend\_handles\_labels()[1] else "")  # Отображение графика  ax2.cla()  ax2.plot(num\_none, label="Неподписанные", c="gray")  ax2.plot(num\_telegram, label="Telegram", c="blue")  ax2.plot(num\_tv, label="TV", c="red")  ax2.plot(num\_x, label="X", c="green")  ax2.set\_xlabel("День")  ax2.set\_ylabel("Количество агентов")  ax2.legend(loc='upper right', fontsize=8, framealpha=0.3)  day\_count += 1  if day\_count >= OBSERVATION\_PERIOD:  plt.close()  return  # Создание фигуры и осей  fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))  plt.subplots\_adjust(bottom=0.3)  # Создание ползунка  ax\_slider = plt.axes([0.15, 0.15, 0.65, 0.03])  slider = Slider(ax\_slider, 'Уровень цифровизации', 0.0, 1.0, valinit=digitalization\_level)  # Создание кнопки  ax\_button = plt.axes([0.8, 0.05, 0.1, 0.04])  button = Button(ax\_button, 'Санкции', hovercolor='0.975')  # Обработчик ползунка  def update\_slider(val):  global digitalization\_level  digitalization\_level = slider.val  slider.on\_changed(update\_slider)  # Обработчик кнопки  def toggle\_sanctions(event):  global sanctions\_active  sanctions\_active = not sanctions\_active  button.label.set\_text('Санкции (вкл)' if sanctions\_active else 'Санкции (выкл)')  button.on\_clicked(toggle\_sanctions)  # Создание анимации  ani = animation.FuncAnimation(fig, update, interval=100, save\_count=OBSERVATION\_PERIOD)  plt.show() |