МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт Информационных Технологий

Кафедра Вычислительной Техники

Отчет по практическим работам №1-4
по дисциплине
«Многоагентное моделирование»

Студент группы: ИКБО-15-22 — Оганнисян Г.А.
(Ф.И.О. студента)

Преподаватель — Гололобов А.А.
(Ф.И.О. преподавателя)

Москва 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире технологии моделирования становятся неотъемлемой частью исследований комплексных систем и процессов. Особое внимание привлекает многоагентное моделирование, которое эффективно применяется для анализа и оптимизации систем с множеством взаимодействующих агентов, оказывающих влияние на общий результат. В данных работах мы обращаемся к созданию многоагентной системы в среде программирования AnyLogic с целью моделирования распространения, использования какого-то процесса, учитывая различные параметры.

AnyLogic представляет собой мощный инструмент для многоагентного моделирования, который позволяет создавать сложные системы и модели, учитывая различные факторы и взаимодействия между агентами. Наша работа направлена на разработку моделей, которые способны анализировать и симулировать различные процессы. Данный подход позволяет более глубоко и точно исследовать динамику инфекционных процессов и выявить факторы, влияющие на их эффективное контролирование.

1. МНОАГЕНТНАЯ МОДЕЛЬ

Для начала было необходимо изучить основные принципы работы в среде AnyLogic. Также нужно определить основные компоненты многоагентного моделирования. В соответствии с названием – нужны агенты, также нужно определить параметры модели, которые необходимы нам для исследования.

1.1 Построение многоагентной модели использование маркетплейсов.

В рамках первой практики была выбрана модель использования маркетплейсов. Параметрами для данной модели будут: эффективность маркетплейса, частота контакта с людьми, вероятность начать использовать маркетплейс, задержка для принятия человеком решения, использовать ли маркетплейс. Была создана модель для 500 агентов, где каждый агент представлял собой личность, у которой соответственно есть 5 состояния: «не использует», «потенциальный пользователь», «использует OZON», «использует WB», «использует Yandex».

Диаграмма состояний показана на следующем рисунке (Рисунок 1.1).

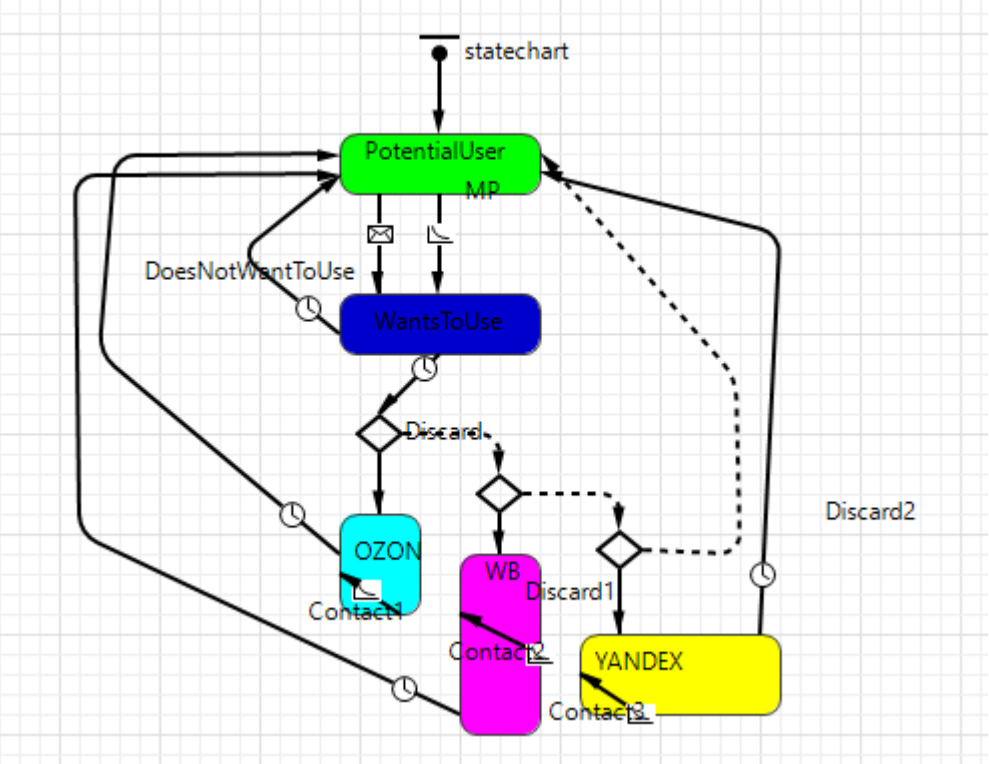

Рисунок 1.1 – Диаграмма состояний личности

Также были использованы следующие параметры для контроля модели (Рисунок 1.2). MPEffectiveness определяет скорость перехода от человека не использующего маркетплеймы к человеку, использующему, ContactRate как часто люди взаимодействуют друг с другом, AdoptionFraction определяет вероятность, того что человек пассивно начнет использовать маркетплейс. DiscardTime – задержку для принятия решения человеком, использовать ли маркетплейс.

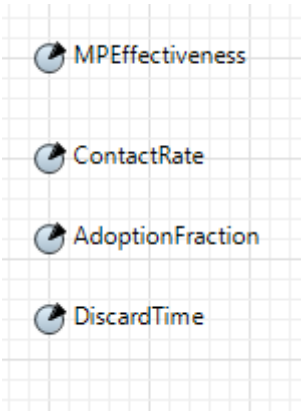

Рисунок 1.2 – Параметры для контроля диаграммы состояний

Для динамического контроля параметра было добавлено 5 ползунка SatisfactionLevelFromUsingMp с значениями для изменения от 2 до 15 с шагом 1, InfluenceOfOthers с значениями для изменения от 5 до 25 с шагом 1, PopulyarOZON с значениями для изменения от 2 до 15 с шагом 1, PopulyarWB с значениями для изменения от 2 до 15 с шагом 1 и PopulyarYandex с значениями для изменения от 2 до 15 с шагом 1. SatisfactionLevelFromUsingSN определяет уровень удовлетворенности от маркетплейсов, что влияет на увеличение или уменьшение людей пользующихся маркетплейсами. InfluenceOfOthers определяет влияние окружающих на маркетплейсы, что влияет на вероятность использования. PopulyarOZON влияет на уровень популярности использования OZON. PopulyarWB влияет на уровень популярности использования Wildberries. PopulyarYandex влияет на уровень популярности использования YandexMarket. Предположительно поведение будет такое: чем больше значение параметра SatisfactionLevelFromUsingSN и больше значение параметра InfluenceOfOthers тем больше будет скорость и вероятность использования.

В конце концов для визуального отслеживания была добавлена временная диаграмма с накоплением. Результат последних действий показан на следующем рисунке (Рисунок 1.3).

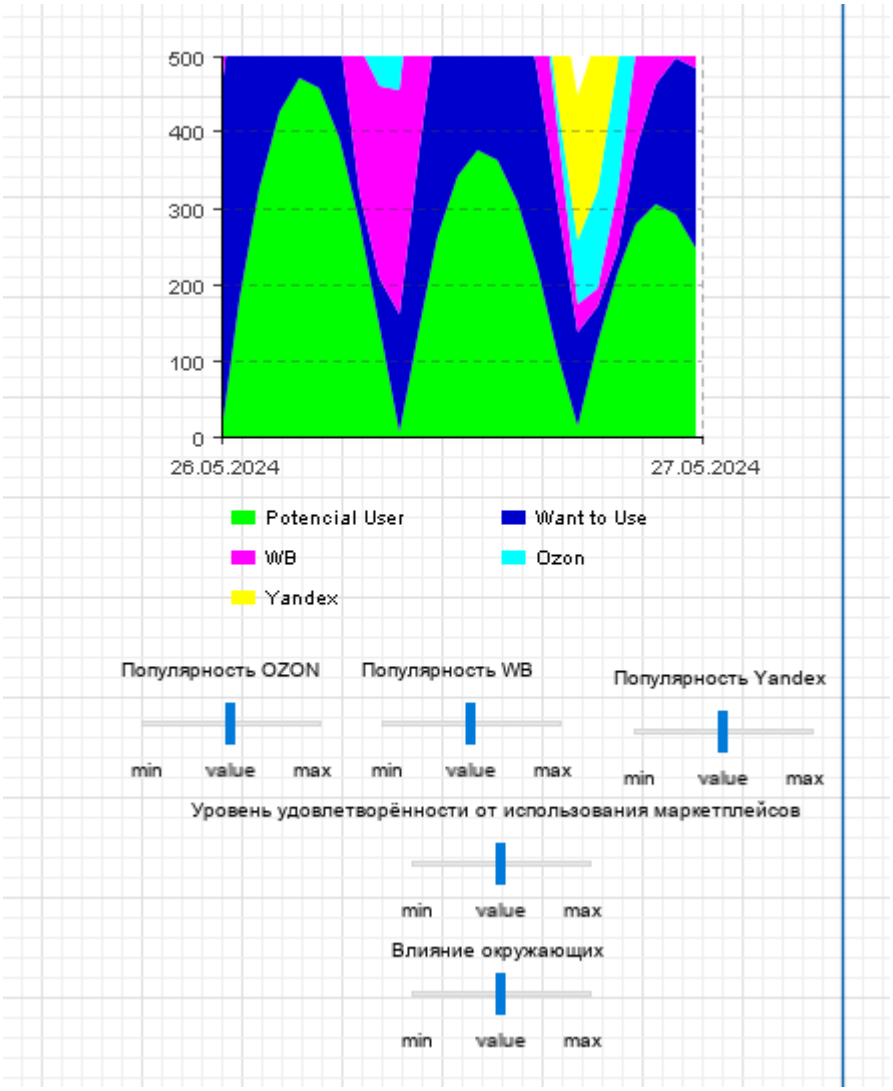

Рисунок 1.3 – Параметры для контроля диаграммы состояний

1.2 Запуск модели использования маркетплейсов для различных сценариев

Для первого запуска было выбрано стандартное значение параметра равное 7 у уровня удовлетворенности, 25 у влияния окружающих, уровень популярности маркетплейсов - случайное число от 2 до 15. Предположительно данная симуляция должна выдать то, что количество использующих людей увеличивается, а количество не использующих уменьшается, но маркетплейсы не будет распространяться быстро. Запустив модель мы получили такой результат (Рисунок 1.4). По нему видно, что модель работает в соответствии с ожиданиями, которые были для неё поставлены.

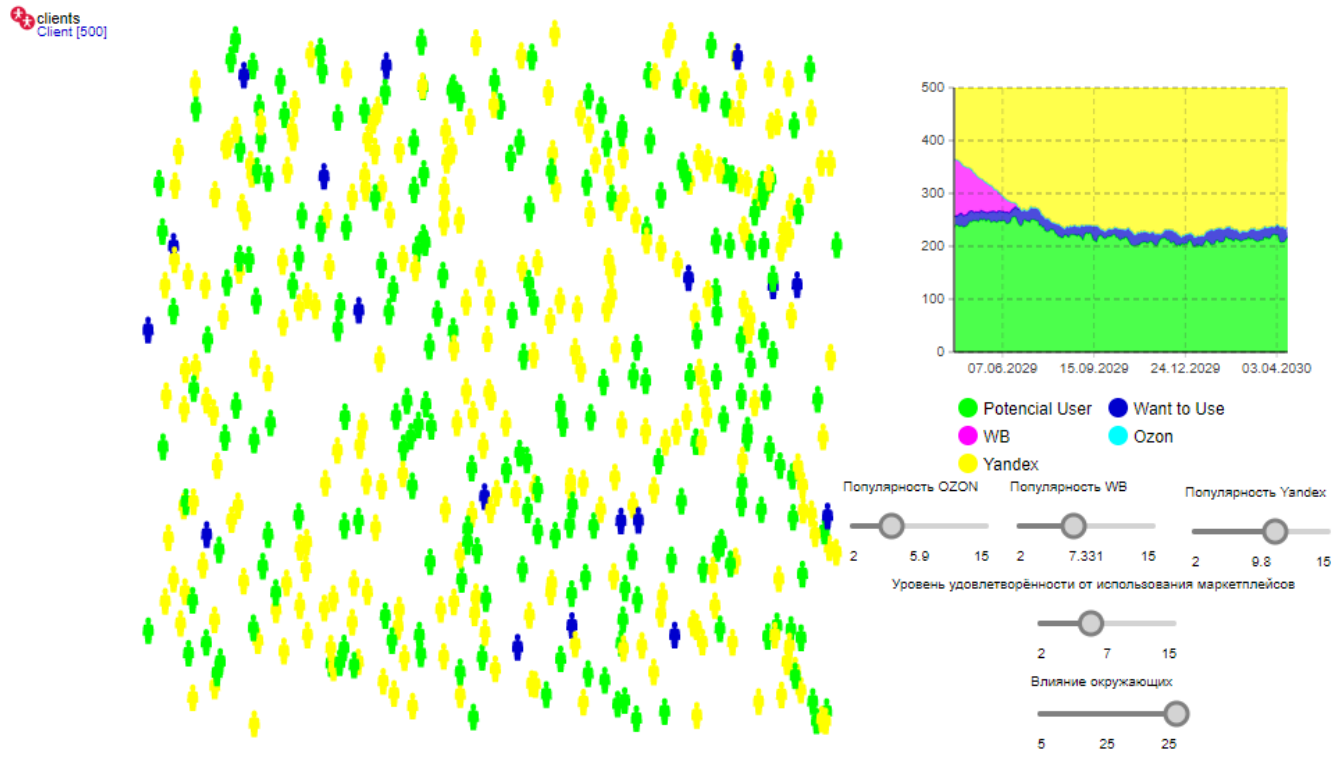

Рисунок 1.4 – Запуск модели при параметре в 7 и 25.

Теперь запустим модель для значения в 15 и 25. (Рисунок 1.5)

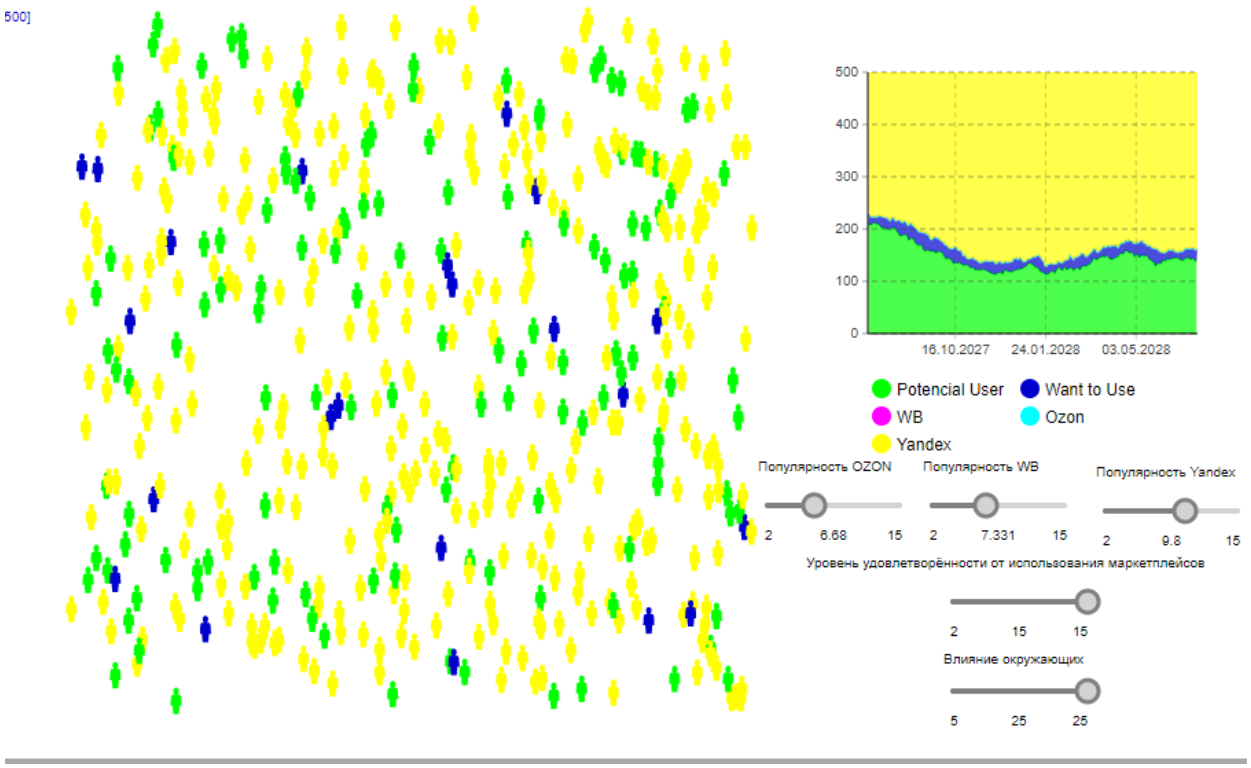

Рисунок 1.5 – Запуск модели при параметре в 15 и 25.

Как мы видим модель также сработала в соответствии с ожиданиями. Теперь же запустим со значениями параметров уровень удовлетворенности 2 и влияние окружающих 25, большинство людей не должны использовать соц.сеть. (Рисунок 1.6)

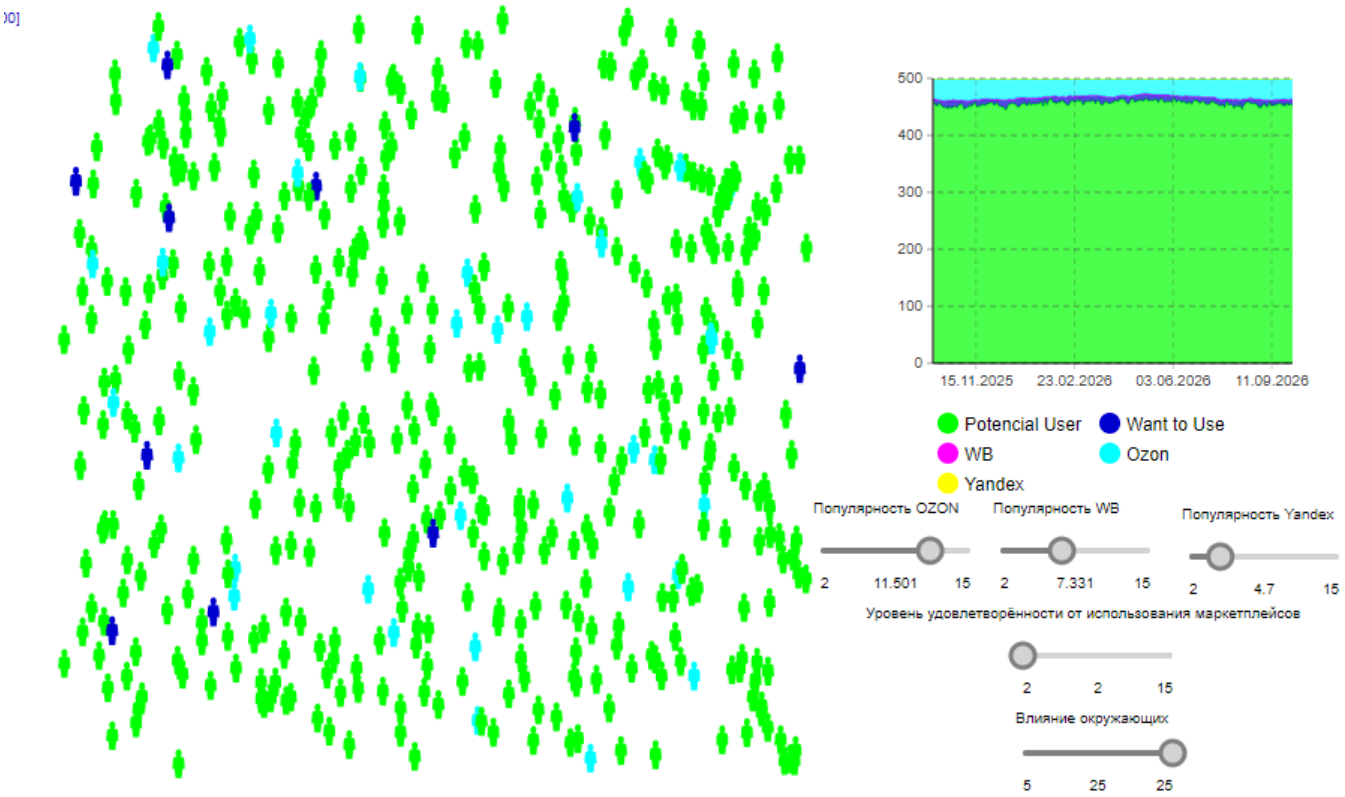

Рисунок 1.6 – Запуск модели при параметре в 2 и 25.

Результаты полностью совпали с ожидаемыми. Таким образом модель была построена верно и отражает действительность.

Запустим модель с параметром популярности Озон больше популярности других маркетпелйсов (Рисунок 1.7). Мы увидим что пользователи пользуются Озон.

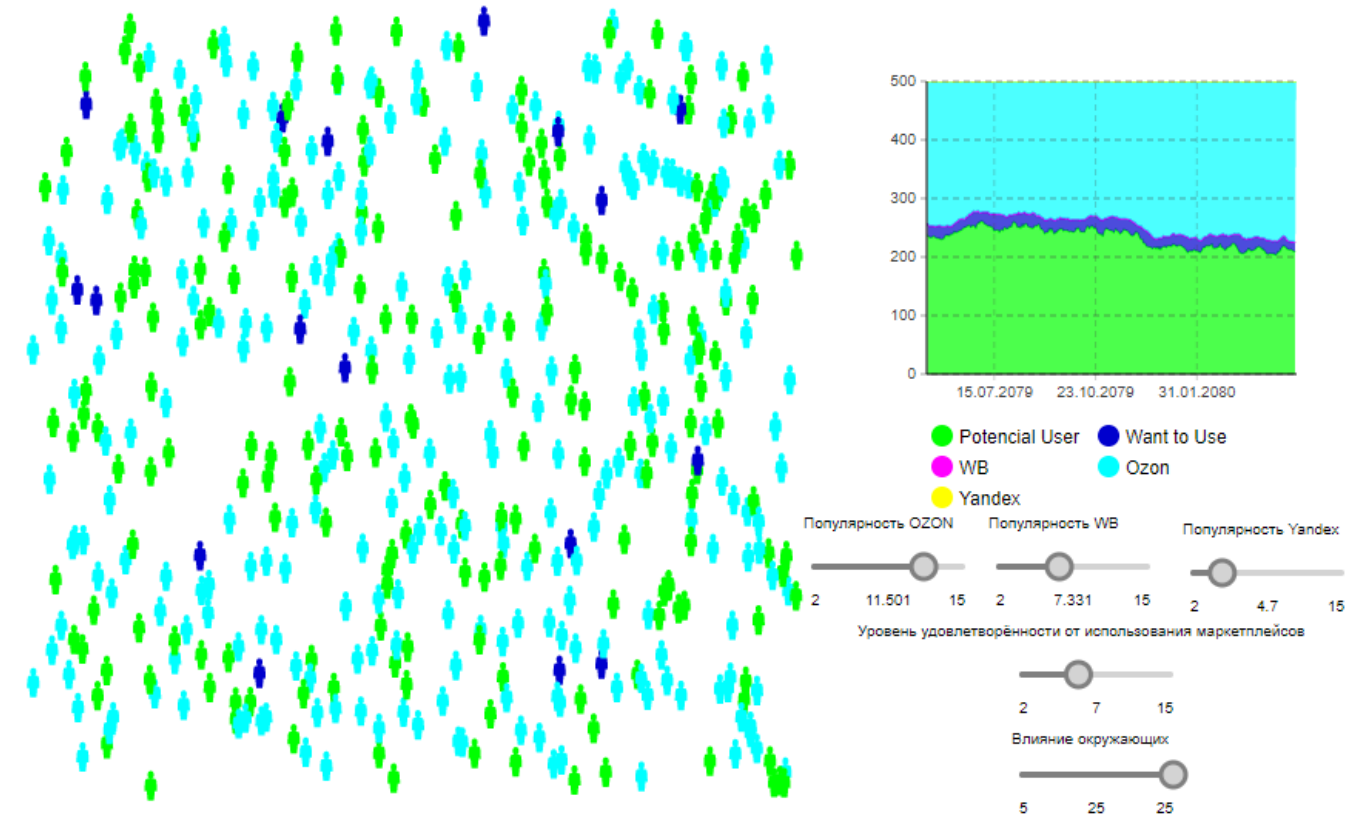

Рисунок 1.7 – Запуск модели завышенной популярности Озон.

Так же аналогично с ЯндексМаркетом (Рисунок 1.8) и Wildberries (Рисунок 1.9)

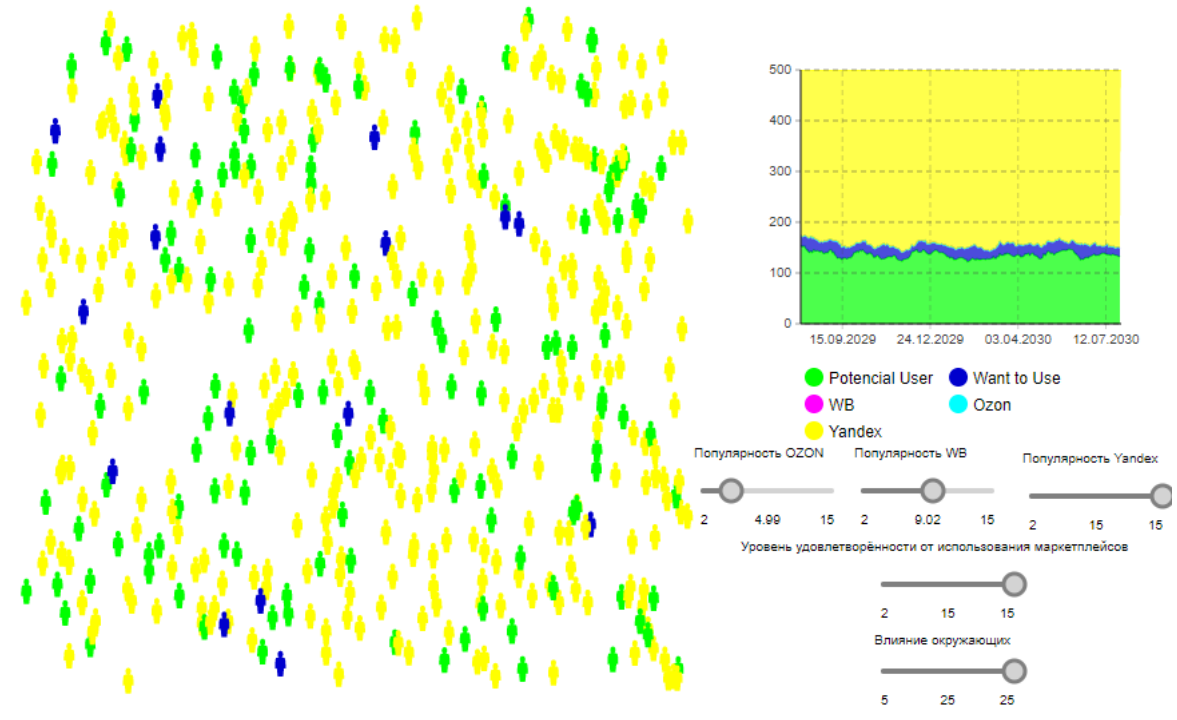

Рисунок 1.8 – Запуск модели завышенной популярности ЯндексМаркета.

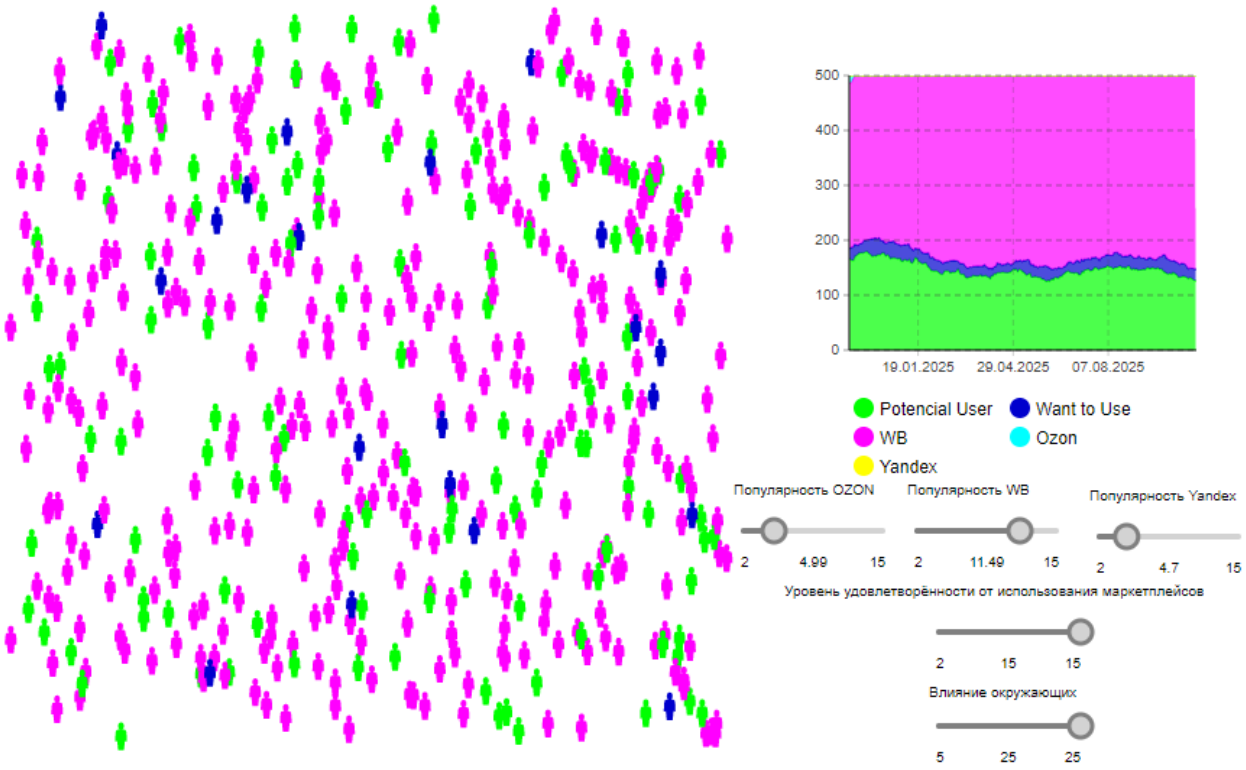

Рисунок 1.9 – Запуск модели завышенной популярности WB.

2. ДИСКРЕТНО-СОБЫТИЙНАЯ МОДЕЛЬ

Необходимо для начала определить основные компоненты системной динамики. Для создания системы переноса товара необходимо создать входную линию и перенос ресурса по всей карте. Так как моделироваться будет передвижение товаров на склад. Нужно определить точку появления товаров и пути переноса, а также точку, куда товары будут переноситься. Параметры для изменения я определил такие: количество паллетов с товаром, количество грузовиков, количество складских рабочих, количество складских разгрузчиков.

2.1 Построение имитационно динамической модели

В рамках второй практической работы была выбрана дискретно событийная модель для рассмотрения моделирование работы склада маркетплейса.

В соответствии с описанием была построена следующая дискретно событийная модель, также добавлены данные для анализа данной модели, графики и ползунки для динамического изменения параметров (Рисунок 2.1).

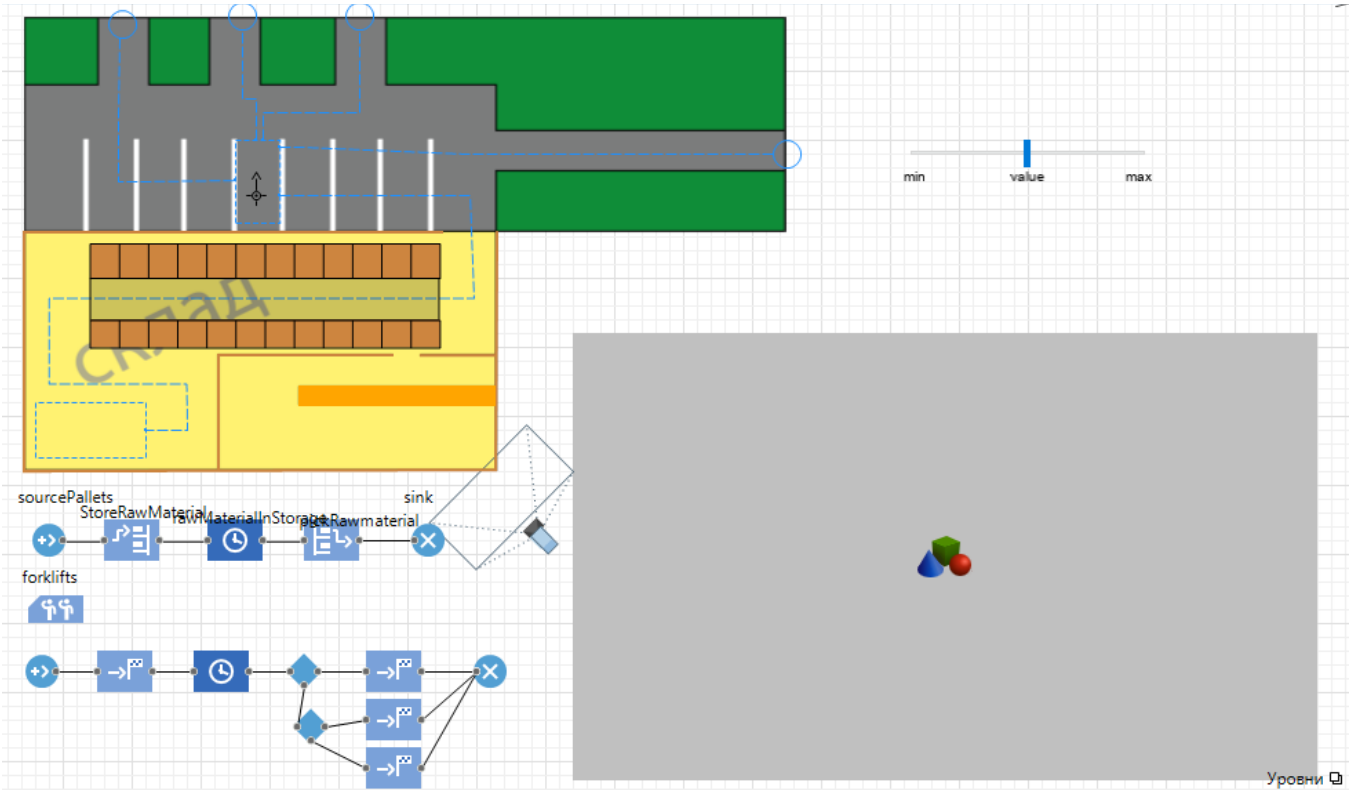

Рисунок 2.1 – Дискретно-событийная модель

Была описана дискретно событийная модель блоками (Рисунок 2.2).

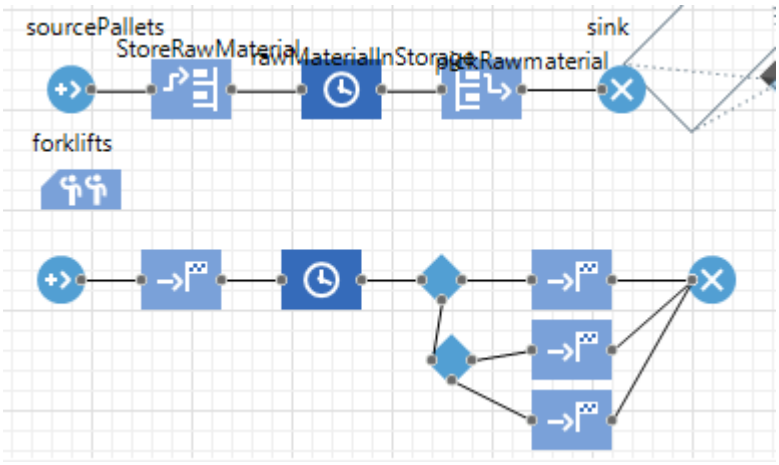

Рисунок 2.2 – Описание дискретно-событийной модели

Также данная модель показывает прибытие паллетов с товарами, а также перенос их по всему складу.

Запустим модель для того, чтобы удостовериться, что она работает. sourcePallets – осуществляет появление паллетов на месте в своей зоне. storeRawMaterial– осуществляет движение разгрузчиков из точки появления в точку появления паллетов с товарами и разгрузку их на стеллажи. Delay – показывает искусственную задержку паллетов на стеллажах, чтобы учесть отсутствие лишнего передвижения погрузчиков, пока паллетов мало. pickRawMaterial – осуществляет передвижение погрузчиков из точки появления в зону со стеллажами и разгрузку товаров со стеллажей в пункт выдачи. sink – осуществляет удаление паллета. Таким образом эта ветка показывает работу склада по разгрузке и плавного перемещения товаров на склад.

Вторая ветка добавляет грузовик, который привозит паллеты с товарами. В данной ветке присутствует ветвление, из-за которого грузовик уезжает по разным путям.

2.2 Запуск модели

Для первого запуска было выбрано значение параметров: количество паллетов с товарами - 15. Запустив модель мы получили такой результат (Рисунок 2.3).

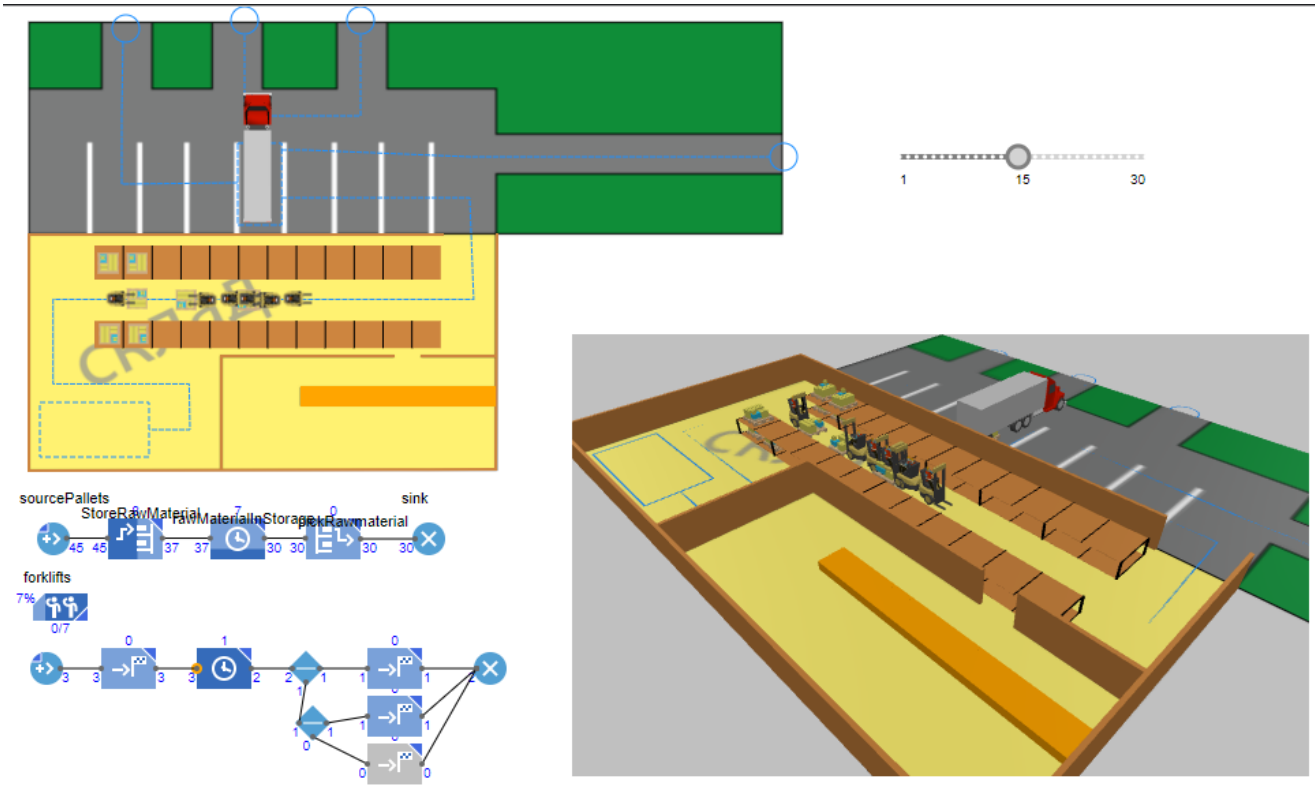

Рисунок 2.3 – Запуск модели при параметре равном 15.

Теперь увеличим количество паллетов до 30 (Рисунок 2.4)

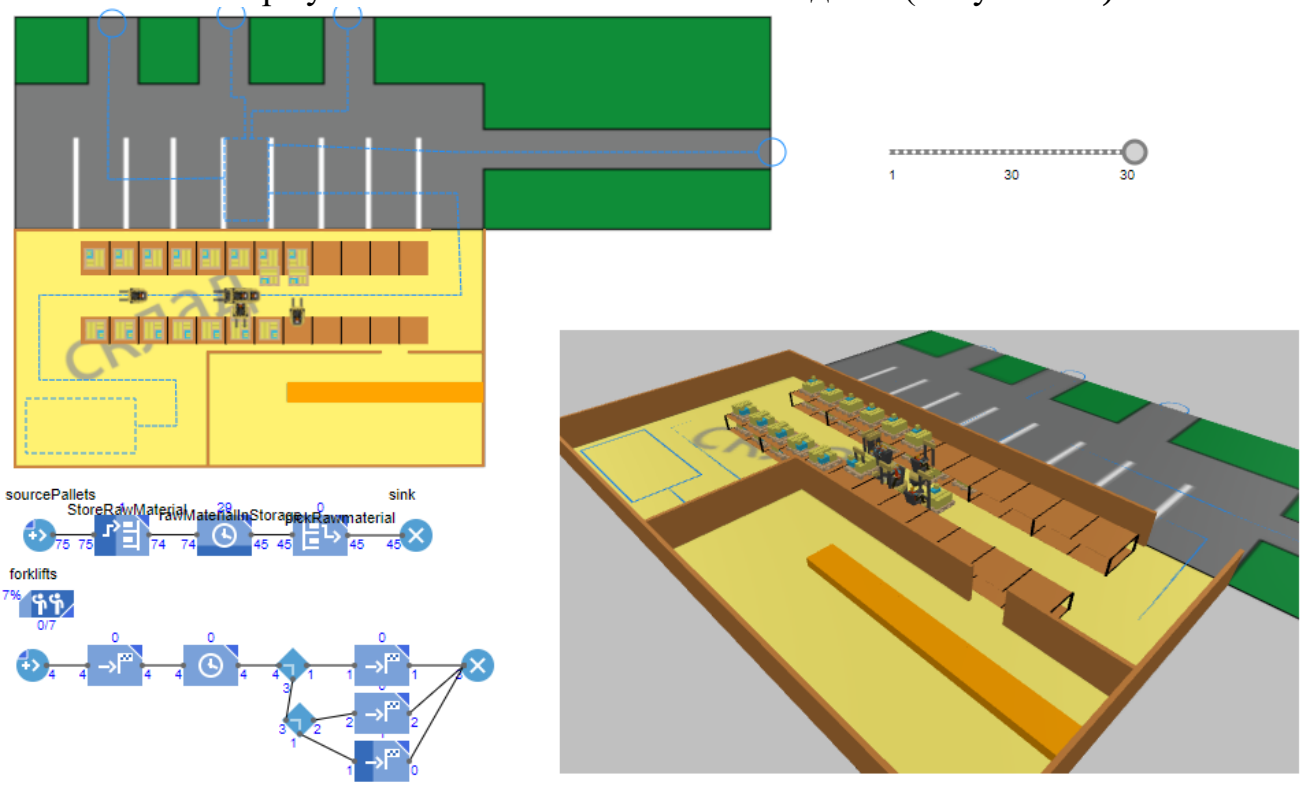

Рисунок 2.4 – Запуск модели при параметре в 30 паллетов.

Как мы видим, что разгрузчики, справляются со своей задачей даже при максимальном значении стартовых трупов. Модель была построена верна и точно показывает действительность

3. МОДЕЛЬ СИСТЕМНОЙ ДИНАМИКИ

Для начала было необходимо изучить основные принципы работы в среде AnyLogic. Также нужно определить основные компоненты системной динамики. Для создания системной динамики нужны накопители и параметры.

3.1 Построение системной динамики

В рамках третьей практической работы была выбрана системная динамика капитала продавцов на маркетплейсе. Параметрами для данной модели будут: общий объем инвестиций, инвестиционная активность, активное время, коэффициент объема торгов, развитие рынка, коэффициент рыночного спроса.

В соответствии с описанием была построена следующая модель системной динамики (Рисунок 3.1). На ней показаны накопители с названиями и параметры, которые влияют на них.

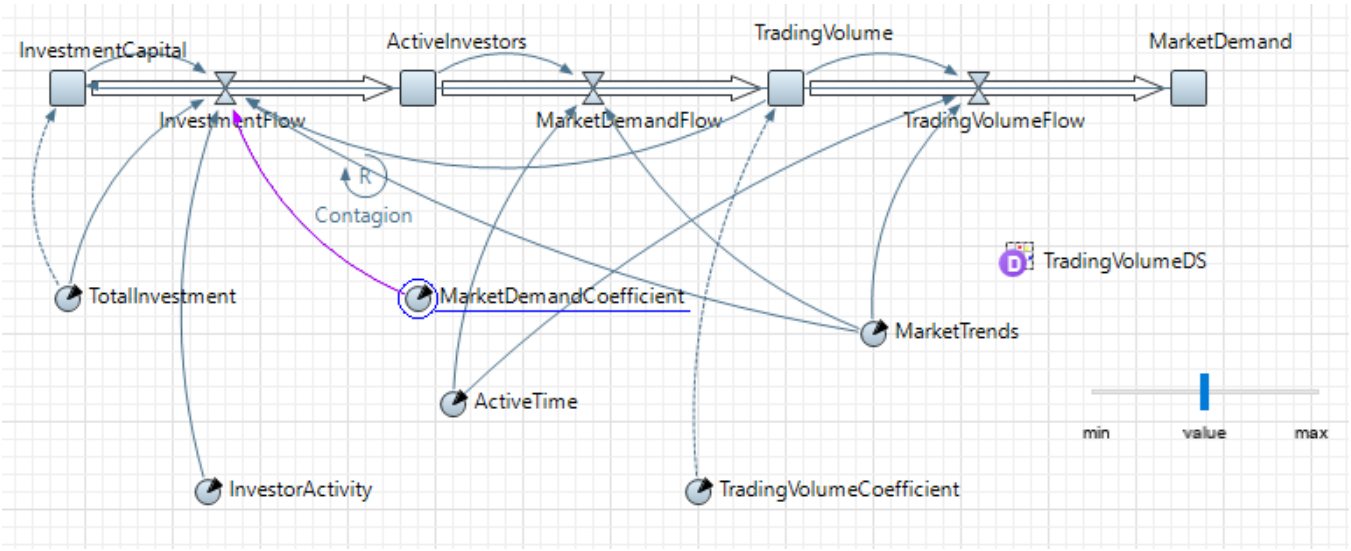

Рисунок 3.1 – Диаграмма системной динамики

Также для контроля изменений были добавлены ползунок и одна диаграмма для контроля накопителей (Рисунок 3.2).

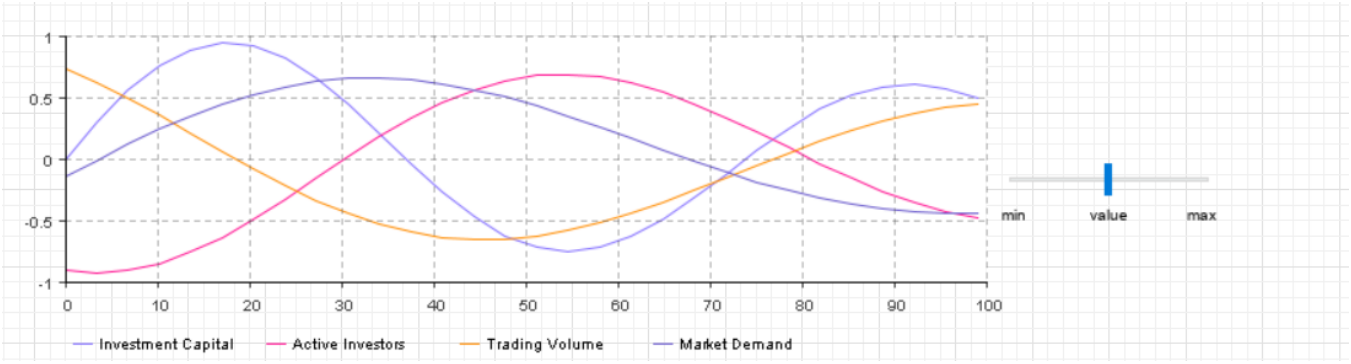

Рисунок 3.2 – Графики и контролирующие элементы

Далее произведём запуск при различных параметрах модели.

3.2 Запуск системной динамики для различных сценариев

Для первого запуска было выбрано следующее значение параметров: поставим значение MarketTrends на 0 с помощью ползунка. Мы увидим что наш капитал никак не будет изменяться, потому что нет развития рынка.(Рисунок 3.3).

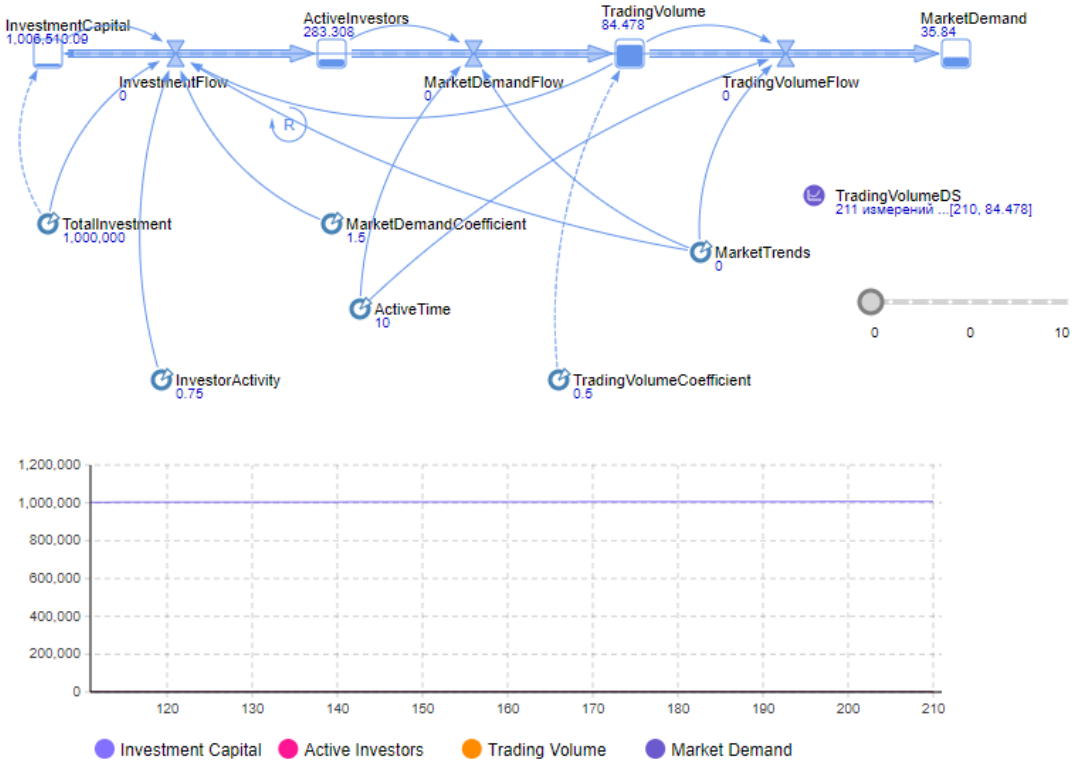

Рисунок 3.3 – Запуск модели при параметрах 0

Теперь увеличим до 8 (Рисунок 3.4)

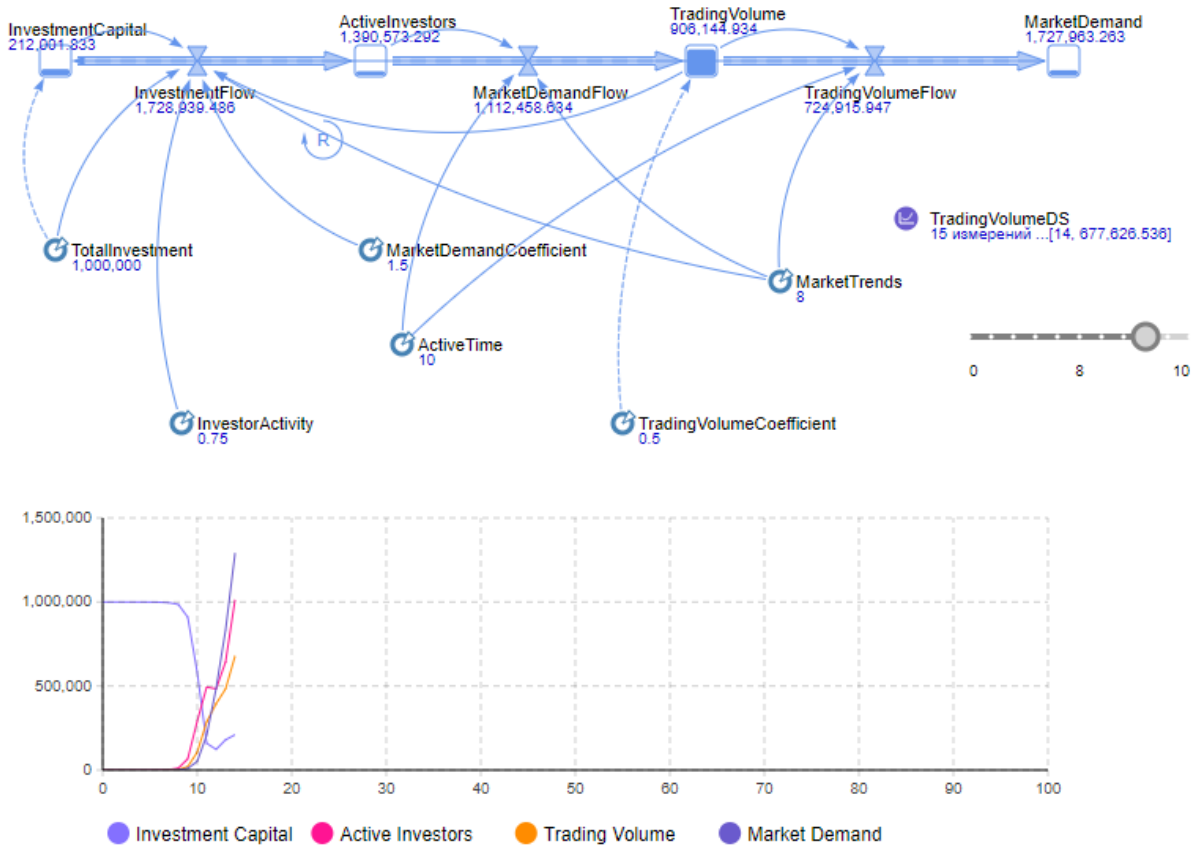

Рисунок 3.4 – Запуск модели при параметре 8

Как мы видим, что при увеличении рыночного спроса у нас активно меняется капитал.

4. ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

Для начала необходимо было изучить различные готовые работы по данной теме. Было изучено множество работ, которые моделировали совершенно разные системы, поэтому мной был сделан вывод, что наилучшим вариантом для создания такой модели будет моделирование с помощью библиотек tkinter и random

4.1 Изучение принципов создания многоагентных систем

Для создания таких систем нужно использовать классы. В моём случае для моделирования распространения вируса были написаны следующие классы: Person и Simulation, причём Person – в нашем случае личность, какой-то человек. Simulation в свою очередь будет использовать агентов и разбрасывать их на поле модели , а так-же задавать параметры заражения наших людей с помощью параметров.

4.2 Практическая часть

4.2.1 Определение с параметрами

Для начала реализации многоагентной системы выбираем параметры. В этой модели я определил уровень опыта продавца на маркетплейсе и его успешность в продажах. Теперь определим состояния модели. В нашем случае это "Новичок", "Опытный" и "Успешный".

4.2.2 Создание класса агента

Д ля реализации класса продавца определяем его поля и методы. Мы уже выбрали параметры выше. Теперь определим методы:

- become_experienced, для изменения состояния с "Новичок" на "Опытный",
- become_successful, для изменения состояния с "Опытный" на "Успешный",
- become_newbie, для изменения состояния с "Опытный" на "Новичок".

В соответствии с описанием был написан код агента модели (Листинг А):

4.2.3 Создание класса модели

Модель будет представлять собой популяцию, состоящую из n-ого количества продавцов. Для инициализации объекта класса модели передаем размер популяции и параметры вероятностей изменения состояний.

Методы данного класса включают:

- promote_random_seller, для продвижения случайного продавца,
- update, для обновления состояния каждого продавца,
- increase_experience_interval и decrease_experience_interval, для увеличения и уменьшения вероятности получения опыта,
- increase_success_interval и decrease_success_interval, для увеличения и уменьшения вероятности достижения успеха,
- run, для запуска основной функции модели.

В соответствии с описанием был написан код модели (Листинг Б):

4.2.4 Основная функция для запуска модели

Одним из требований было создание кнопки, по которой модель будет запускаться. Для этого создаем основную функцию, которая запускает модель, а также вспомогательную, которая вызывается при нажатии на кнопку.
В соответствии с этим был написан код основной функции (Листинг В)

4.2.5 Запуск программы

Теперь запустим нашу модель и проверим, как она работает. После запуска программы появляется окно с продавцами в виде прямоугольников и различные кнопки. Чтобы начать продвижение продавцов, нужно нажать на кнопку «Продвигать продавцов», и случайный продавец станет опытным. (Рисунок 4.2.1)

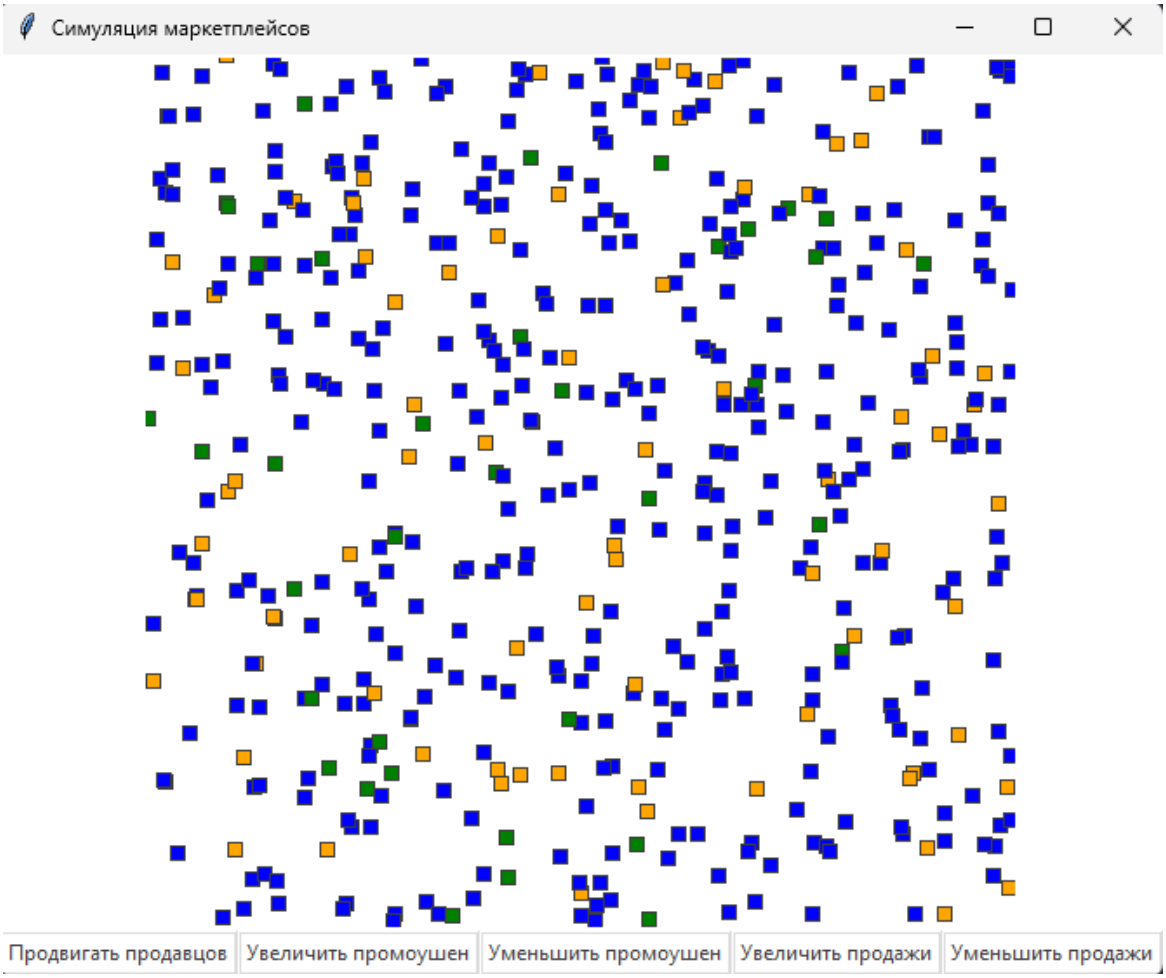

Рисунок 4.2.1 – Прогресс после нажатия кнопки

Сразу после нажатия кнопки начнется продвижение продавцов, и они будут менять свои состояния. Продавцы, достигшие состояния "Опытный", изменят цвет с синего на оранжевый. Продавцы, достигшие состояния "Успешный", изменят цвет с оранжевого на зеленый. Продавцы, которые вернутся в состояние "Новичок", снова станут синими. (Рисунок 4.2.2).

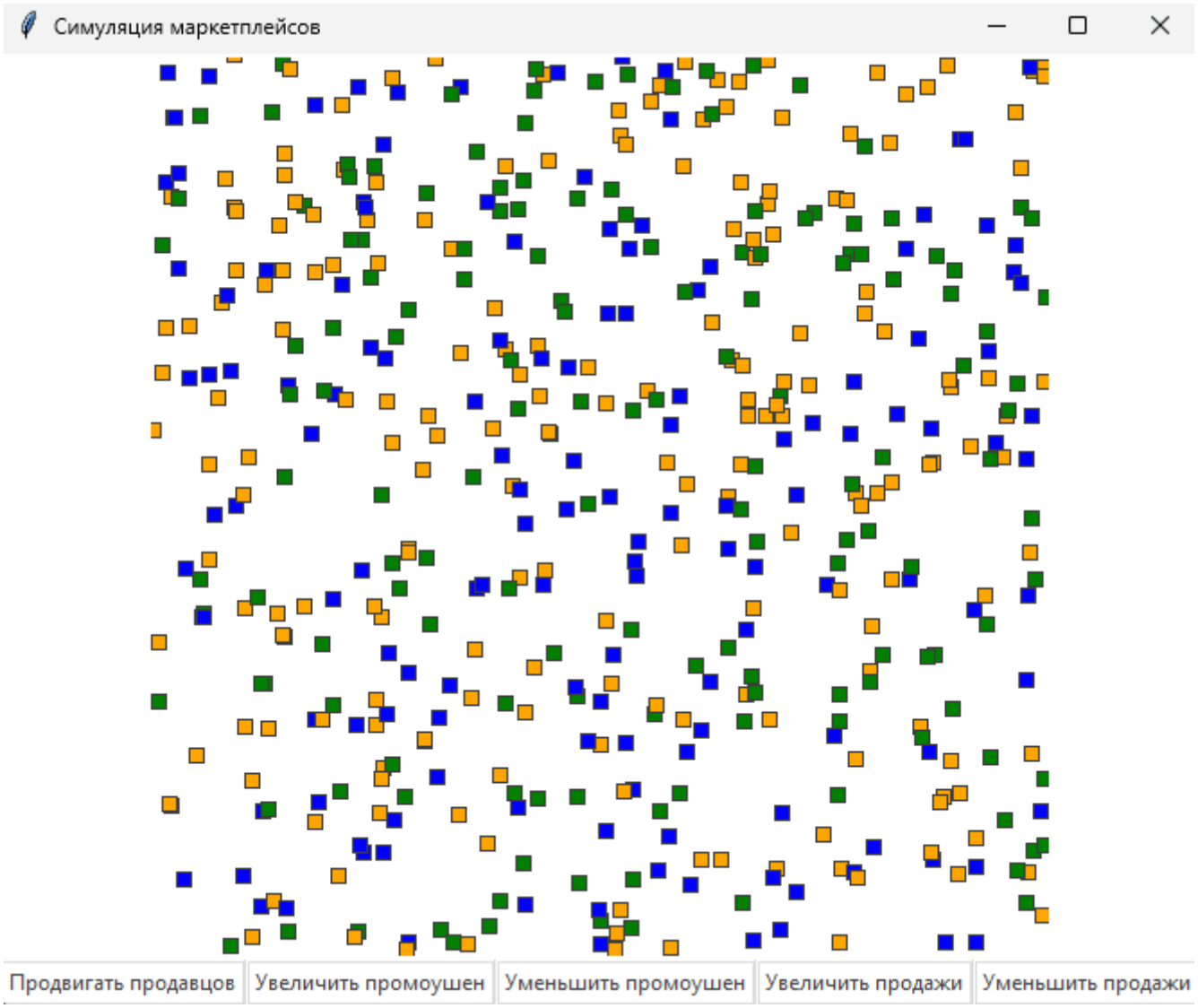

Рисунок 4.2.2 – Прогресс продвижения продавцов

Можно динамически изменять параметры во время работы программы нажатием на соответствующие кнопки. Например, можно увеличить или уменьшить вероятность получения опыта или достижения успеха. Это позволяет гибко управлять процессом моделирования и наблюдать за изменениями в популяции продавцов. (Рисунок 4.2.3).

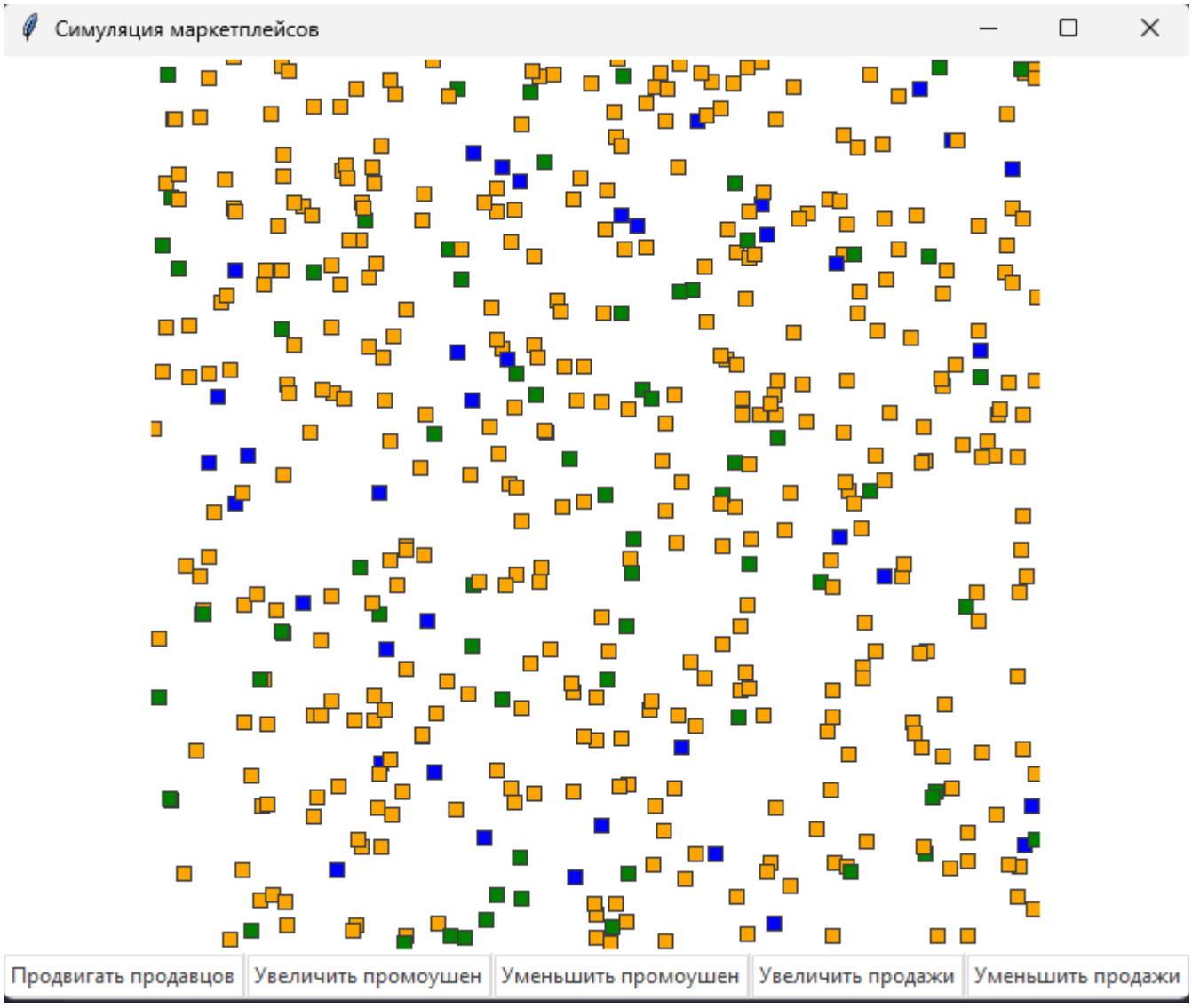

Рисунок 4.2.3 – Динамическое изменение параметров

ЗАКЛЮЧЕНИЕ

В результате практических работ были разработаны различные модели, позволяющие симулировать разнообразные системы.

Были изучены основы создания симуляций в различных средах разработки и на современных языках программирования.

1. Многоагентная модель на языке программирования Python с использованием библиотек tkinter и random:

Эта модель представляет собой симуляцию, где множество агентов взаимодействует друг с другом и с окружающей средой. Каждый агент обладает своими характеристиками и правилами поведения..

2. Модель распространения социальной сети:

В этой модели исследуется распространение социальной сети в популяции, учитывая параметры удовлетворенности от использования социальной сети и влияние окружающих.

3. Дискретно-событийная модель для работы крематория:

В данной модели симулируется работа крематория, представляя систему как набор дискретных событий, происходящих в определенные моменты времени.

4. Модель системной динамики для управления заправкой самолетов:

Эта модель представляет собой аналитическую или компьютерную симуляцию системы управления заправкой самолетов, включая хранилище топлива, баки каждого самолета, параметры выбора заправки, задержку передачи, коэффициент полезного действия и другие.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Официальный сайт AnyLogic [Электронный ресурс]. Создание многоагентных моделей. – Режим доступа: https://www.anylogic.ru/

2. Практические работы РТУ МИРЭА по дисциплине Многоагентное моделирование [Электронный ресурс] / Создание многоагентных моделей. – Режим доступа: https://online-edu.mirea.ru/course/view.php?id=6558

ПРИЛОЖЕНИЯ

Приложение А – Листинг 1

Приложение Б – Листинг 2

Приложение В – Листинг 3

Приложение А

Листинг 1 – Класс агента.

```
import tkinter as tk
import random


class Seller:
    def __init__(self, canvas, x,
y):
        self.canvas = canvas
        self.x = x
        self.y = y
        self.state = "Новичок"
        self.shape =
canvas.create_rectangle(x-4, y-4,
x+4, y+4, fill="blue")

    def become_experienced(self):
        if self.state == "Новичок":
            self.state = "Опытный"

self.canvas.itemconfig(self.shape,
fill="orange")

    def become_successful(self):
        if self.state == "Опытный":
            self.state = "Успешный"

self.canvas.itemconfig(self.shape,
fill="green")

    def become_newbie(self):
        if self.state == "Опытный":
            self.state = "Новичок"
            self.canvas.itemconfig(s
            elf.shape, fill="blue")
```

Приложение Б

Листинг 2 – Класс модели.

```
class MarketplaceSimulation:
    def __init__(self, width, height, num_sellers, experience_interval,
success_interval, demote_interval, return_interval):
        self.width = width
        self.height = height
        self.num_sellers = num_sellers
        self.experience_interval = experience_interval
        self.success_interval = success_interval
        self.demote_interval = demote_interval
        self.return_interval = return_interval
        self.sellers = []
        self.experienced_count = 0

        self.root = tk.Tk()
        self.root.title("Симуляция маркетплейсов")
        self.canvas = tk.Canvas(self.root, width=width, height=height,
bg="white")
        self.canvas.pack()

        self.promote_button = tk.Button(self.root, text="Продвигать продавцов",
command=self.promote_random_seller)
        self.promote_button.pack(side=tk.LEFT)

        self.increase_experience_button = tk.Button(self.root, text="Увеличить
промоушен", command=self.increase_experience_interval)
        self.increase_experience_button.pack(side=tk.LEFT)

        self.decrease_experience_button = tk.Button(self.root, text="Уменьшить
промоушен", command=self.decrease_experience_interval)
        self.decrease_experience_button.pack(side=tk.LEFT)

        self.increase_success_button = tk.Button(self.root, text="Увеличить
продажи", command=self.increase_success_interval)
        self.increase_success_button.pack(side=tk.LEFT)

        self.decrease_success_button = tk.Button(self.root, text="Уменьшить
продажи", command=self.decrease_success_interval)
        self.decrease_success_button.pack(side=tk.LEFT)

        for _ in range(num_sellers):
            x = random.randint(0, width)
```

Продолжение листинга Б.

```
            y = random.randint(0, height)
            seller = Seller(self.canvas, x, y)
            self.sellers.append(seller)

    def promote_random_seller(self):
        for seller in self.sellers:
            if seller.state == "Новичок":
                if random.random() < self.experience_interval:
                    seller.become_experienced()
                    self.experienced_count += 1

        self.root.after(100, self.promote_random_seller)

    def update(self):
        if self.experienced_count > self.num_sellers / 2:
            self.success_interval = 0.1

        for seller in self.sellers:
            if seller.state == "Опытный":
                if random.random() < self.success_interval:
                    seller.become_successful()
                    self.experienced_count -= 1
                elif random.random() < self.demote_interval:
                    seller.become_newbie()
                    self.experienced_count -= 1
            elif seller.state == "Успешный":
                if random.random() < self.return_interval:
                    seller.state = "Новичок"
                    seller.canvas.itemconfig(seller.shape, fill="blue")
                    self.experienced_count -= 1

        self.root.after(100, self.update)

    def increase_experience_interval(self):
        self.experience_interval *= 1.1

    def decrease_experience_interval(self):
        self.experience_interval *= 0.9

    def increase_success_interval(self):
        self.success_interval *= 0.8

    def decrease_success_interval(self):
        self.success_interval *= 1.2

    def run(self):
        self.update()
        self.root.mainloop()
```

Приложение В

Листинг В – Параметры симуляции и запуск программы.

```
# Параметры симуляции
WIDTH = 500
HEIGHT = 500
NUM_PEOPLE = 500
satisfaction_level = 0.1
contact_rate = 0.8

sim = Simulation(WIDTH, HEIGHT, NUM_PEOPLE)
sim.run()
```