

ДИСЦИПЛИНА	Программные средства имитационного моделирования систем (полное наименование дисциплины без сокращений)
ИНСТИТУТ	ИТ
КАФЕДРА	Прикладной математики полное наименование кафедры)
ВИД УЧЕБНОГО МАТЕРИАЛА	Практики (в соответствии с пп.1-11)
ПРЕПОДАВАТЕЛЬ	Есипов Иван Владимирович (фамилия, имя, отчество)
СЕМЕСТР	7, 2024-2025 (указать семестр обучения, учебный год)



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

Учреждение высшего образования

МИРЭА – Российский технологический университет

Институт Информационных Технологий

Кафедра Прикладной математики

Практическая работа №10

Тема практической работы

«Расширенная модель павильона метро»

Москва 2024

Смоделируем работу станции метро с применением комбинации нескольких библиотек: применим моделирование пешеходных потоков, движение поездов по рельсам, работу турникетов, эскалаторов, имитации погрузки пассажиров в поезд и высадку из поезда новоприбывших.

Шаг 1. Создание каркаса станции

Начнем с создания физического представления станции – ее структуры, уровней и платформ. Рассмотрим станцию, имеющую 2 входа и совместные 2 выхода. Каждая пара вход-выход оснащена собственной платформой, находящейся на определенном уровне и оборудована турникетами для прохода и автоматами по продаже билетов.

Пройдя турникеты, люди с помощью эскалаторов перемещаются на платформу посадки и высадки с поездов, принадлежащую другому уровню. Платформа содержит стены, имитирующие разницу в размерах станции и длиной поезда, и естественные ограждения, применяемые для контроля и разграничения пассажиропотока.

Этап 1. Создание платформы

Создадим платформу как набор стен, ограничивающих передвижение пассажиров. В палитре пешеходной библиотеки выберем элемент «Стена» и выстроим контур платформы:

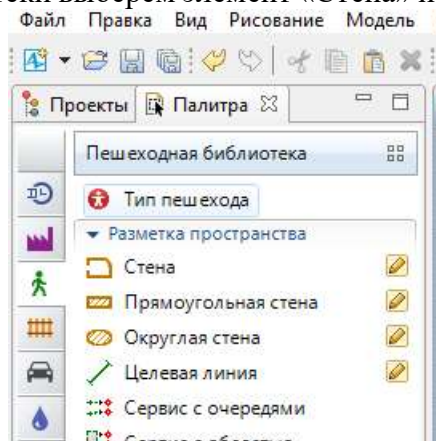


Рисунок 283. Палитра

Платформа имеет 2 подхода к поездам – они должны быть широкими, чтобы пассажиры могли поступить в любой вагон приезжающего поезда, и 2 выхода – для контакта с эскалаторами. Так же введем невысокие стены-разделители пассажиропотоков. Предлагаемый план станции выглядит так:

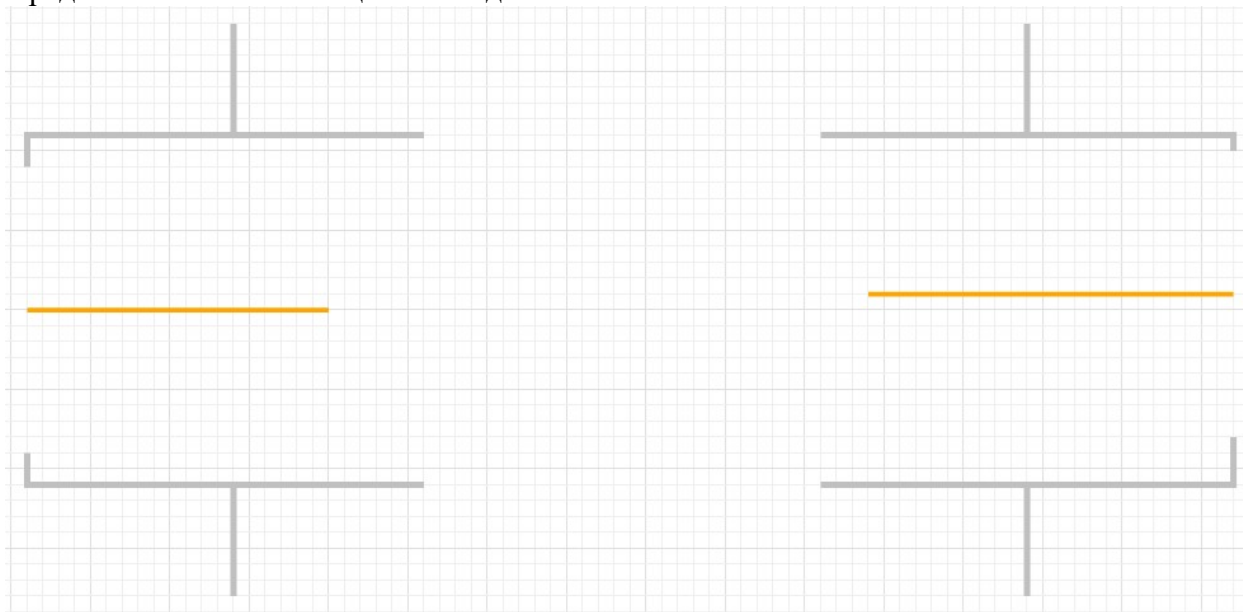


Рисунок 284. План станции

Серые стены описывают естественные стены станции, оранжевые – ограничители потоков. В свойствах серых стен укажем параметр «Z-высота» равным 40. Высоту оранжевых стен возьмем по 20.

Этап 2. Создание блоков для входа и выхода пассажиров

Сделаем зал разделенным на 2 секции – секцию входа и выхода. Так же сделаем зал не прямоугольной формы, а с геометрией, ориентированной на широкий входной поток людей с возможным отходом за покупкой билетов.

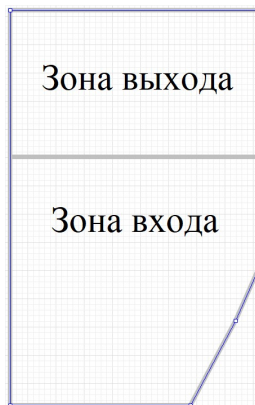


Рисунок 285. Зоны входа и выхода

В настройках свойств стены укажем высоту 40 пикселей. Разделяющую стену по аналогии с платформой сделаем высотой 20.

Этап 3. Включение и настройка эскалаторов. Настройка уровней

Будем считать, что от входа после прохода турникета пассажир будет спускаться на платформу посредством эскалатора. Эскалатор в Anylogic – инструмент перехода между разными уровнями модели. Добавим в модель 4 группы эскалаторов из палитры библиотеки пешеходов – по 1 на спуск и подъем на каждую платформу входов/ выходов.

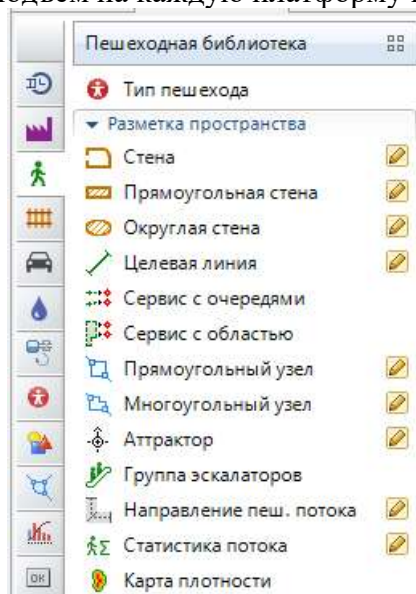


Рисунок 286. Палитра

Включим и настроим элементы. В поле свойств эскалатора можно настраивать уровни, между которыми происходит переход по эскалаторам группы. Включим в каждую группу по 3 эскалатора, задав необходимое значение в поле «Кол-во эскалаторов». Параметр «Задает наклон как» установим в положение «подъем» – таким образом, нам достаточно указать высоту, реализующую переход, а в модели программа автоматически согласует наклон эскалатора в зависимости от длины и высоты. Настройку поведения пассажиров можно оставить по умолчанию, либо организовать по своему желанию – будут пешеходы идти по ступеням, либо будут стоять.

Для добавления нового уровня достаточно просто кликнуть по полю «уровень» или «верхний уровень» и нажать «новый уровень». Создайте и настройте новый уровень, назвав его «ground1».

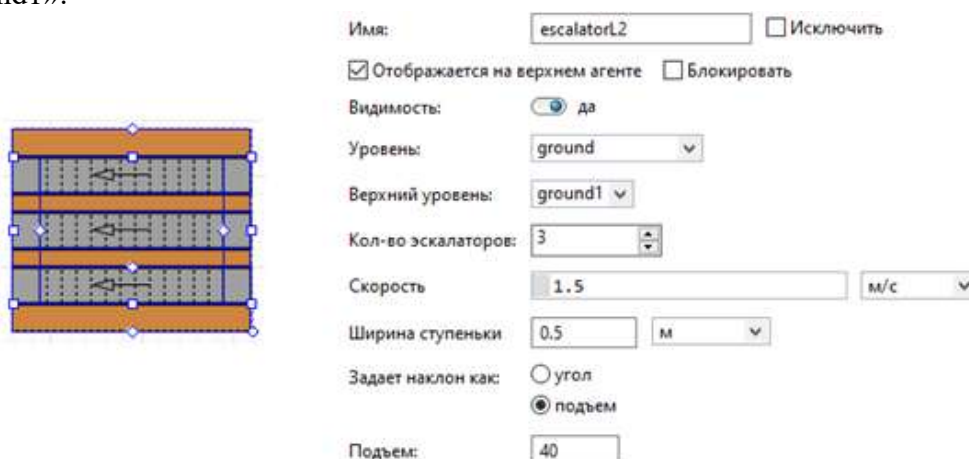


Рисунок 287. Настройка эскалаторов.

Внутри каждой группы эскалаторов можно настраивать направление движения каждой ленты отдельно, однако в нашей модели мы не будем заниматься настройкой разнонаправленных эскалаторов – для простоты автоматического управления эскалаторами.

Для правильной работы системы присвоим всем элементам платформ входа-выхода (стенам) значение поля «уровень» в свойствах «местоположение и размер» значение, указанное в поле «верхний уровень» группы эскалаторов.

Таким образом, на данном этапе мы создали разноуровневую структуру станции.

Шаг 2. Создание и организация железнодорожной подсистемы.

Этап 1. Введение структуры

Поработаем над разметкой железнодорожного полотна. Поезда поступают на станцию с двух сторон в разных направлениях движения. Не будем делать сложную цепь, соединяющую несколько станций – по окончании этой работы вы сможете это сделать самостоятельно.

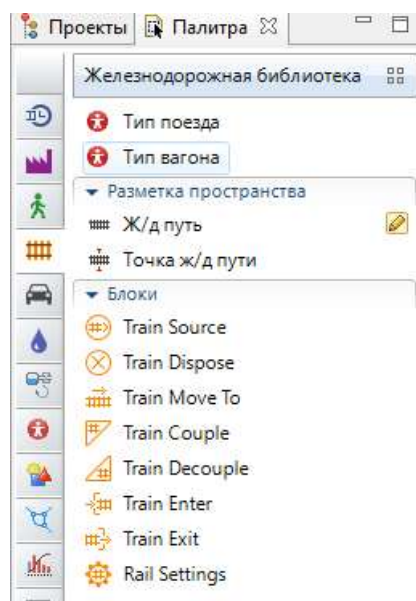


Рисунок 288. Палитра

Добавим в модель 2 элемента из железнодорожной библиотеки: «Ж/д путь». Их отрисовка подобна отрисовке элемента «Стена». Проведем их вдоль нашей станции:

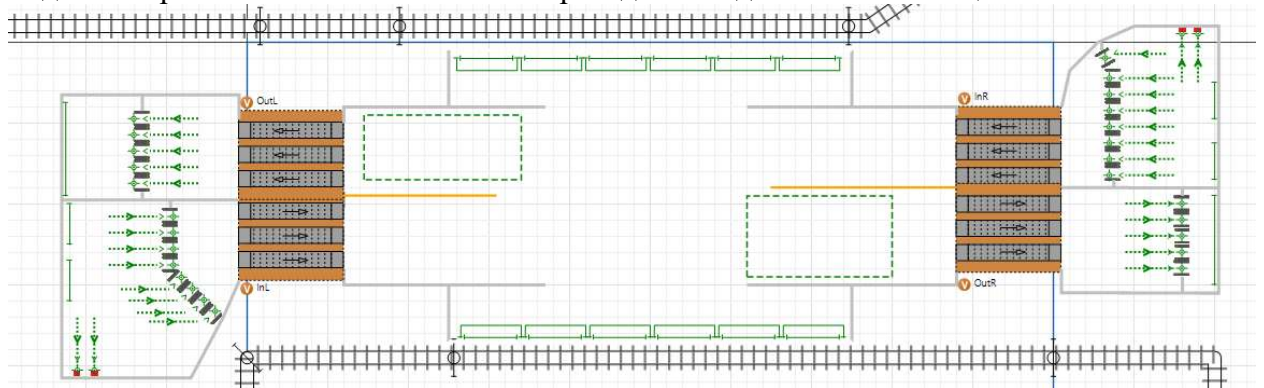


Рисунок 289. Железные дороги.

Пока что на зеленые элементы можно не обращать внимание – они относятся к разметке пространства, регулирующей пешеходные потоки. На пути для управления ж/д транспортом нужно добавить по 3 элемента «Ж/д узел» – исходный; элемент, регламентирующий положение остановки для высадки; конечный пункт. Возможна ситуация, в которой поезд в будущем будет генерироваться или исчезать некорректно, поэтому рекомендуется не экономить пространство и длину полотна – можно построить длинный путь, и поставить старт и конец вдали от самой станции.

Этап 2. Построение логической схемы ж/д подсистемы

Добавим в модель элементы из Ж/д библиотеки: Source, MoveTo, Dispose, а так же элементы Delay из библиотеки моделирования процессов. Соберем 2 цепочки для каждой линии поездов: возникновение поезда, его движение до остановки, задержка на разгрузку, задержка для загрузки пассажиров, отъезд от станции и исчезновение поезда. Примерный вид цепочек представлен ниже:

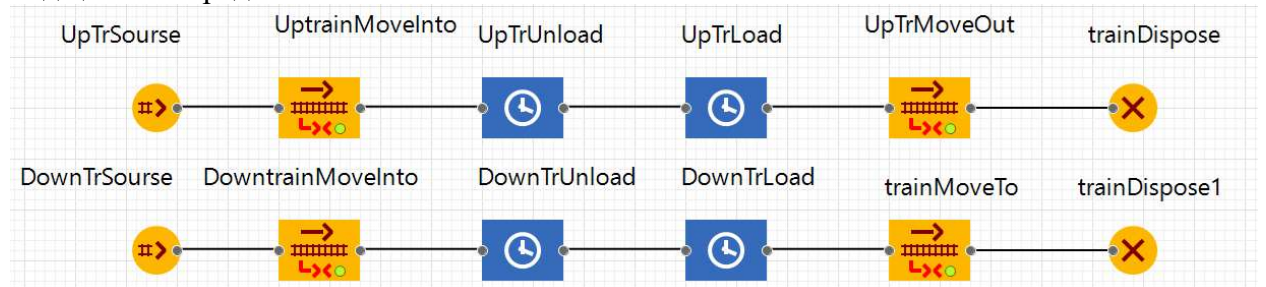


Рисунок 290. Логическая схема

Этап 3. Настройка логической схемы ж/д подсистемы

Соотнесем положения на схеме логическим элементам Ж/д подсистемы. Для блоков «Source» в свойствах установим следующее состояние:

Поезда прибывают согласно:	=	Времени между прибытиями	▼
Время между прибытиями:	↻	uniform(35, 70)	секунды ▼
Первое прибытие происходит:	=	после заданного времени	▼
Считать параметры агентов из БД:	=	<input type="checkbox"/>	
Кол-во прибытий ограничено:	=	<input type="checkbox"/>	
Кол-во вагонов (включая локомотив):	↻	3	





После создания:	=	Поместить поезд на путь узла	▼
Точка входа задается как:	=	<input checked="" type="radio"/> Точка ж/д пути <input type="radio"/> Смещение на пути	
Точка ж/д пути:	=	pointOnTrack1	▼  
Направление на пути:	=	Вперед (первый вагон ближе к концу пути)	▼

Рисунок 291. Настройка блоков Source

Таким образом, мы задаем подачу поездов – время между прибытиями будет случайной величиной, равномерной между 35 и 70 секундами, один поезд будет состоять из 3 вагонов, и возникать в точке Ж/д пути – в узле, который мы создавали ранее. В блоках «MoveTo» Ж/д библиотеки укажем пункт назначения – узел, находящийся возле станции для первого блока в каждой цепи, и узел, являющийся выходным – для второго.

Имя:	UptreinMoveInto	<input checked="" type="checkbox"/> Отображать имя	<input type="checkbox"/> Исключить
Направление движения:	=	<input checked="" type="radio"/> Вперед <input type="radio"/> Назад	
Маршрут:	=	Не задан (поезд будет следовать согласно стрелкам)	▼

Цель движения:	=	Заданная точка пути	▼
Точка ж/д пути:	=	pointOnTrack	▼  

Крейсерская скорость:	↻	0	м/с ▼
При начале движения:	=	Ускорять/тормозить до крейсерской скорости	▼

Рисунок 292. Настройка блоков MoveTo

В блоках задержки настроим следующие свойства:

Имя:	UpTrUnload	<input checked="" type="checkbox"/> Отображать имя	<input type="checkbox"/> Исключить
Тип задержки:	=	<input checked="" type="radio"/> Определенное время <input type="radio"/> До вызова функции stopDelay()	
Время задержки:	↻	15	секунды ▼
Вместимость:	=	1	
Максимальная вместимость:	=	<input type="checkbox"/>	


Место агентов:	=		▼ 
----------------	---	--	---

Рисунок 293. Настройка блоков Delay

В нашей модели не очень важно место агентов, эти узлы выполняют функции временной задержки – 15 секунд на разгрузку и 7 – на загрузку пассажиров (числа приняты из собственных соображений автора модели, вам позволитется назначить меньшие или большие значения, что может привести к дестабилизации работы модели). Помимо этого, они реализуют функционал, регулирующий потоки пассажиров, с которым разберемся

дальше. Таким образом, мы организовали движение поездов – возникновение, подъезд к станции, задержка, отъезд от станции и исчезновение.

Шаг 3. Организация и настройка пешеходной подсистемы

Этап 1. Анализ модели и логики движения пешехода на станции

Рассмотрим логику движения человека на станции: существуют 2 типа пешеходов станции – те, кто прибыл на станцию снаружи, и идёт на поезд, и те, кто выходит с поезда и выходит в нужном ему направлении. Логика их действий близка, но имеет несколько различий. Движение пассажира, следующего на поезд, представляет следующую последовательность:

- 1) Возникновение на входном холле;
- 2) С некоторой вероятностью у пассажира возникает потребность в покупке проездного билета;
- 3) Все пассажиры, владеющие проездным билетом, проходят турникеты (что занимает различное время на извлечение, билета, его приложение и считывание);
- 4) Пройдя турникет, пассажир спускается на эскалаторе, и, исходя из собственных знаний, выбирает необходимый ему Ж/д путь, и вагон для посадки;
- 5) Проходит и встаёт к определенной области в ожидании поезда, а по прибытию вагона исчезает со станции.

Движение пешехода, сошедшего с поезда, описывается более простой схемой:

- 1) Он выходит из вагона, в котором находился в процессе движения поезда;
- 2) Определяет из собственных соображений необходимый выход;
- 3) Проходит по эскалаторам;
- 4) Проходит через турникеты, и выходит со станции.

Исходя из описанной логики, можно понять, что в нашей модели не хватает блоков-турникетов и автоматов по продаже билетов, блока удержания пешехода в ожидании поезда, а так же входов и выходов, работающих с пешеходами. За это отвечают элементы разметки пространства пешеходной библиотеки: «целевая линия» (для входов и выходов) и «сервис с очередями» – для турникетов и автоматов.

Для возникновения пассажиров используется блок «Ped Source», для исчезновения – «Ped Sink». Для моделирования процесса выбора будем применять блок «Ped Select Output». Основным механизмом движения пешеходов является блок «Ped Go To», ожидания – «Ped Wait», а прохождение турникета и покупка билета – «Ped Service».

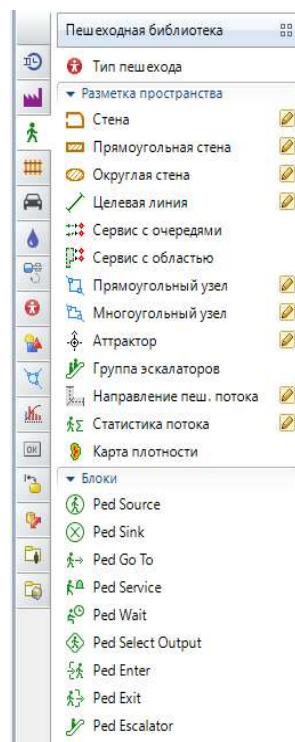


Рисунок 294. Палитра

Этап 2. Внедрение некоторых дополнительных элементов пешеходной библиотеки

На основе проведенного анализа добавим в модель объекты разметки пространства пешеходной библиотеки так, чтобы реализовать схему: вход на станцию осуществляется с двух дверей (целевые линии), возле которых стоят автоматы по продаже билетов (сервис с очередями), за которыми следуют турникеты (сервис с очередями). После спуска на эскалаторе пешеход следует к целевой линии ожидания, после которой он следует к выходу из модели.

Структура пешеходных блоков входного холла на примере левого зала выглядит так:

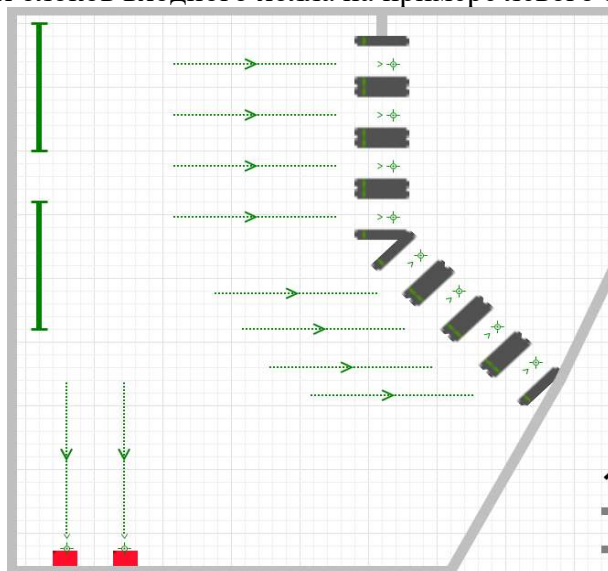


Рисунок 295. Пешеходные блоки входного холла

2 зеленые линии слева сверху – входные целевые линии. Зеленые стрелки с крестообразными аттракторами относятся к блокам «сервис с очередями». Для управления этим блоком можно нажать один раз на блок, после чего отдельным кликом выбрать необходимый аттрактор или линию очереди и переставить/повернуть по своему желанию:

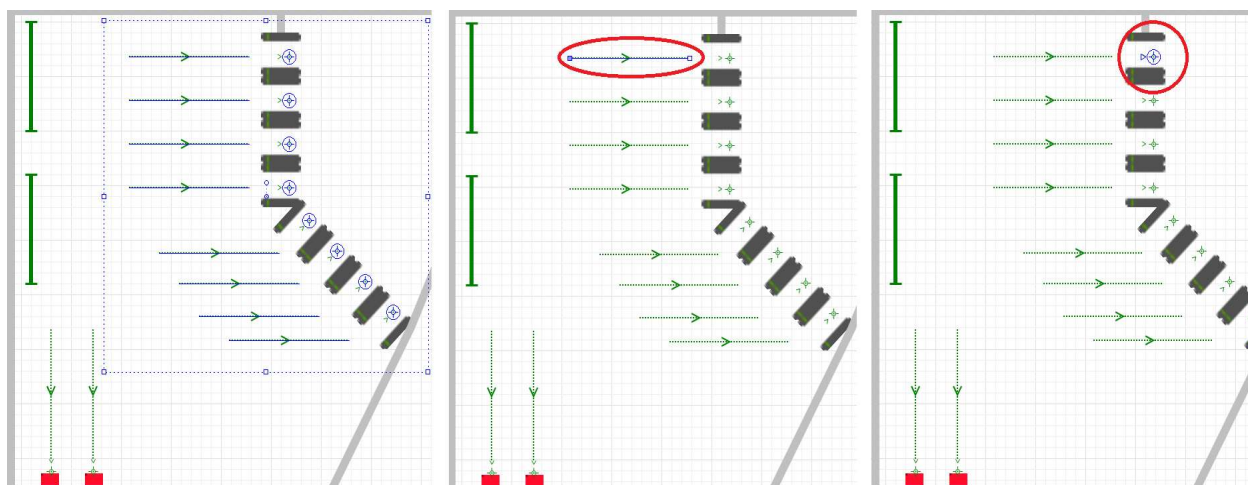


Рисунок 296. Пешеходные блоки входного холла

В свойствах сервиса турникетов настроим следующие параметры: количество сервисов, количество очередей, уровень.

services - Сервис с очередями

Имя: ☐ Исключить ☒ Отображается на верхнем агенте

☐ Блокировать

Видимости: ☒ да

Уровень:

Кол-во сервисов:

Кол-во очередей:

Тип очереди: ☒ Линия ☐ Змейка

Тип сервиса: ☒ Точечный ☐ Линейный

Кол-во обратных очередей:

Обслуживать пешеходов из:

- ☐ Самой длинной очереди
- ☒ Ближайшей очереди
- ☐ Ближайшей непустой очереди
- ☐ Следующей очереди (по порядку)
- ☐ Очереди с приоритетом
- ☐ Очередь задается пользователем

Рисунок 297. Настройка турникетов

Для блоков продажи билетов установим следующие настройки:

TicketsLeft - Сервис с очередями

Имя: ☐ Исключить ☒ Отображается на верхнем агенте

☐ Блокировать

Видимость: ☒ да

Уровень:

Кол-во сервисов:

Кол-во очередей:

Тип очереди: ☒ Линия ☐ Змейка

Тип сервиса: ☒ Точечный ☐ Линейный

Кол-во обратных очередей:

Обслуживать пешеходов из:

- ☐ Самой длинной очереди
- ☒ Ближайшей очереди
- ☐ Ближайшей непустой очереди
- ☐ Следующей очереди (по порядку)
- ☐ Очереди с приоритетом
- ☐ Очередь задается пользователем

Рисунок 298. Настройка блоков продажи билетов

Красные блоки снизу – модельки автоматов по продаже билетов, множество черных параллельных блоков – турникеты. Их можно добавить в модель для наглядности открыв панель «3D объекты», вкладку «Аэропорт».

Так мы организовали структуру входных холлов, теперь организуем пешеходную структуру платформы на примере платформы нижней линии поездов:

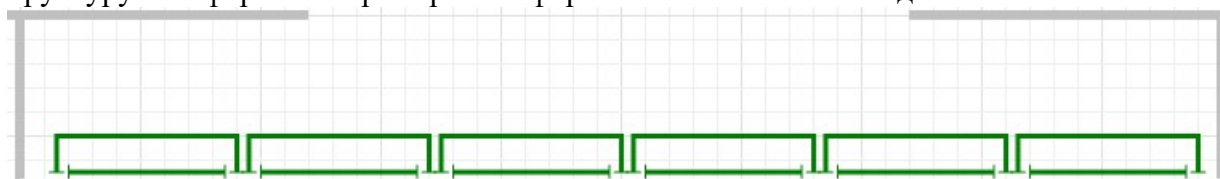


Рисунок 299. Пешеходная структура платформы

Для 3 вагонов введем 6 линий ожидания (полукруглые), и соответствующие им целевые линии выхода (прямые). Для выходящих из вагонов пешеходов эти прямые целевые линии будут, очевидно, начальными. Дополнительная настройка для целевых линий не требуется. Для упрощения управлением эскалаторами введем дополнительные области, которые будут фиксировать подходящих к эскалатору на выход со станции пешеходов: добавим блоки «прямоугольный узел» библиотеки пешеходов перед эскалаторами.

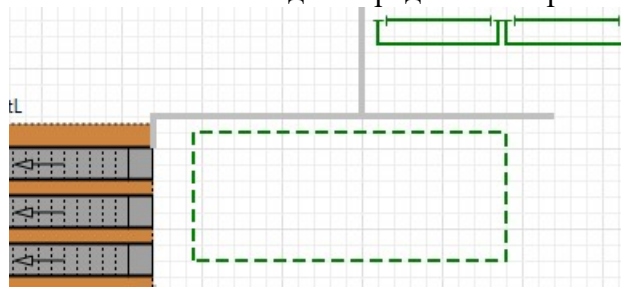


Рисунок 300. Область перед эскалаторами

Структура выходного холла на примере левого зала предлагается следующей:

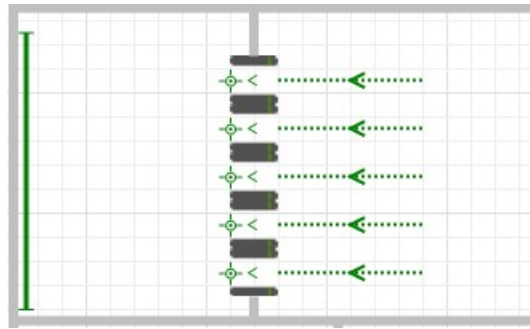


Рисунок 301. Структура выходного холла

Единая выходная целевая линия и предшествующие ей 5 турникетов – сервисов со следующей настройкой:

services3 - Сервис с очередями

Имя: ☐ Исключить ☒ Отображается на верхнем агенте

☐ Блокировать

Видимость: ☒ да

Уровень:

Кол-во сервисов:

Кол-во очередей:

Тип очереди: ☒ Линия ☐ Змейка

Тип сервиса: ☒ Точечный ☐ Линейный

Кол-во обратных очередей:

Обслуживать пешеходов из:

- ☐ Самой длинной очереди
- ☒ Ближайшей очереди
- ☐ Ближайшей непустой очереди
- ☐ Следующей очереди (по порядку)
- ☐ Очереди с приоритетом
- ☐ Очередь задается пользователем

Рисунок 302. Настройка турникетов

Таким образом, наша модель выглядит следующим образом:

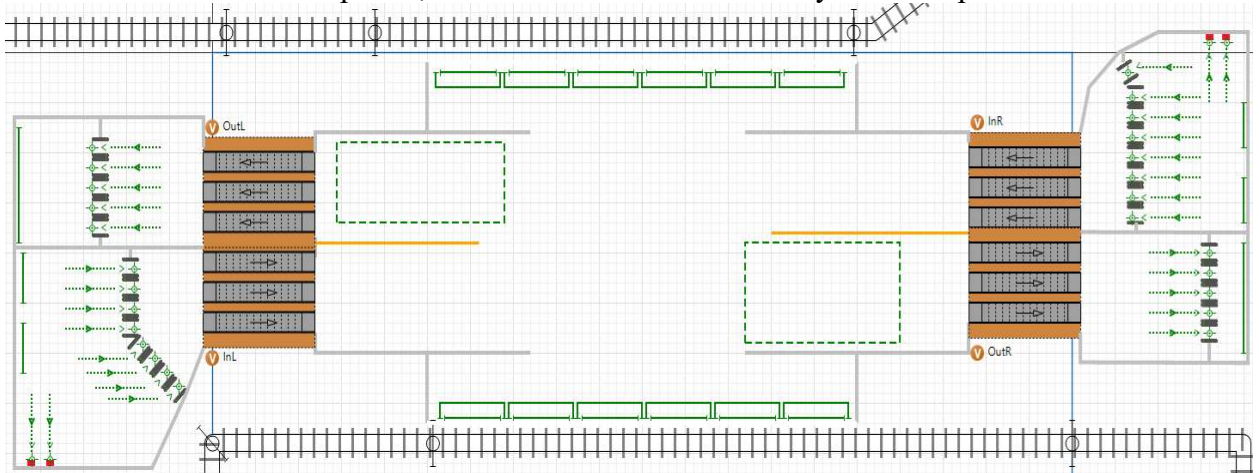


Рисунок 303. Общий вид модели

В качестве дополнительного визуализатора модели добавим «карту плотности» пешеходной библиотеки.



Рисунок 306. Блоки PedSource

Каждый блок соответствует своей начальной целевой линии. Настроим связи с линиями:

Имя: IncomingR1 ☒ Отображать имя ☐ Исключить

Место появления: ☒ линия ☐ точка (x, y) ☐ область

Целевая линия:

Прибывают согласно:

Интенсивность:

Количество прибытий ограничено: ☐

Рисунок 307. Настройка блоков PedSource

Выбрать линию можно либо из списка, либо указанием на конкретную линию курсором. Возникшему в холле пассажиру предлагается выбор между проходом сразу к турникетам, либо сначала к автоматам по продаже билетов. Реализуем это с применением модуля Ped Select Output:

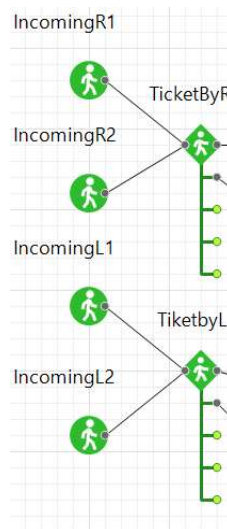


Рисунок 308. Распределение потоков людей

Привязки к разметке пространства этот блок не имеет. Настроим вероятности на выходах блока:

Имя:

Использовать: ☒ Вероятности ☐ Условия ☐ Номер выхода

Козфф. предпочтения 1:

Козфф. предпочтения 2:

Рисунок 309. Настройка блоков Ped SelectOutput

Соответственно, будем считать, что каждый 10-ый человек не имеет билета. Им надо пройти в Ped Service:

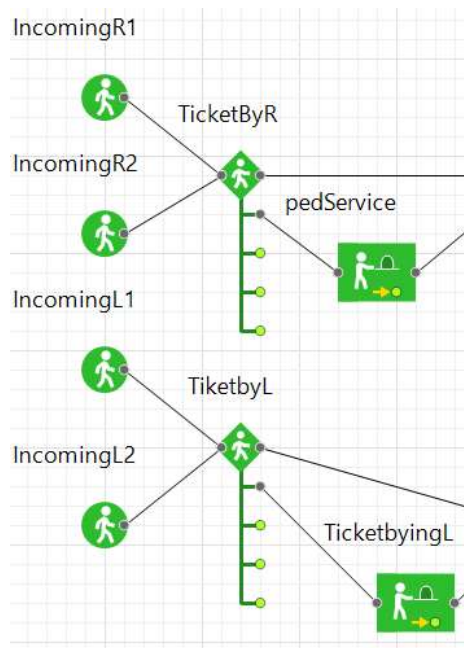


Рисунок 310. Блоки PedService

В настройках сервиса продажи билетов введем следующие параметры: очередь (привязка к разметке пространства) и время задержки.

Имя: ☒ Отображать имя ☐ Исключить

Сервисы:

Выбирается очереди:

Время задержки: секунды

Задержка на восстановление: секунды

Проходить в обратном направлении: ☐

Рисунок 311. Настройка блоков PedService

Соответственно, будем считать, что для покупки билета человеку потребуется от 12 до 30 секунд. После приобретения билета пешеходы проходят турникеты pedService1 и 2:

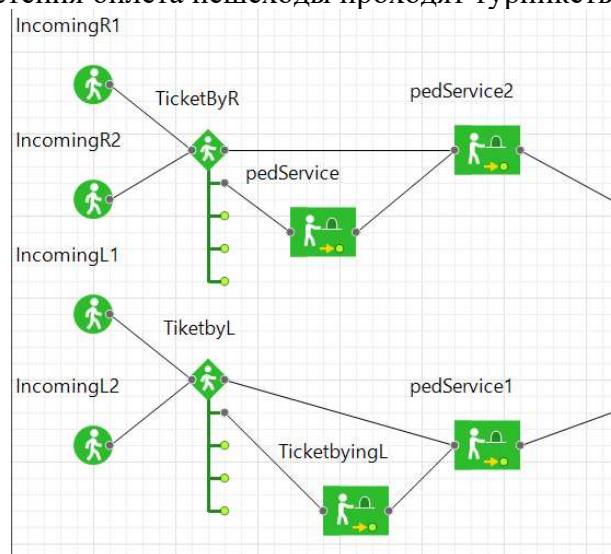


Рисунок 312. Блоки PedService

В настройках зададим те же условия, что для покупки билетов, но будем считать, что для прохода потребуется от 0.1 до 0.7 секунд.

После прохождения турникетов пешеход попадает на эскалатор. Для этого в логическую схему включаем блок `pedEscalator`:

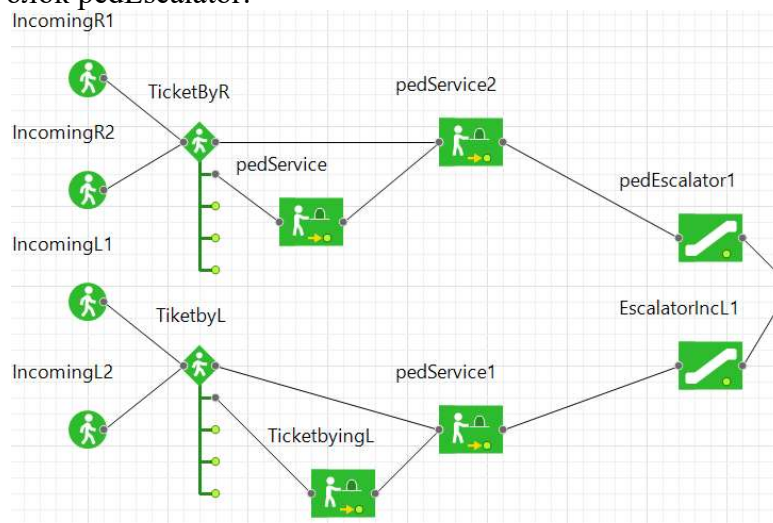


Рисунок 313. Блоки `pedEscalator`

В настройках нового блока привязываем его к элементу разметки пространства и задаем направление прохождения – вниз:

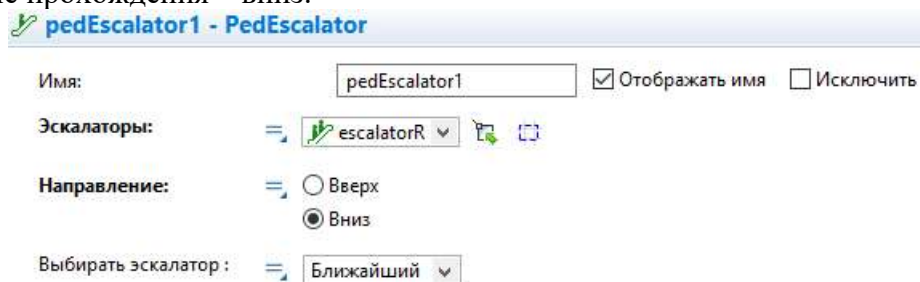


Рисунок 314. Настройка блоков `pedEscalator`

Оба эскалатора спускают пассажиров к одной платформе, на которой пешеходы принимают решение – идти к одному поезду, или ко второму. Такой выбор реализуем ещё одним блоком `Ped Select Output` с равными вероятностями:

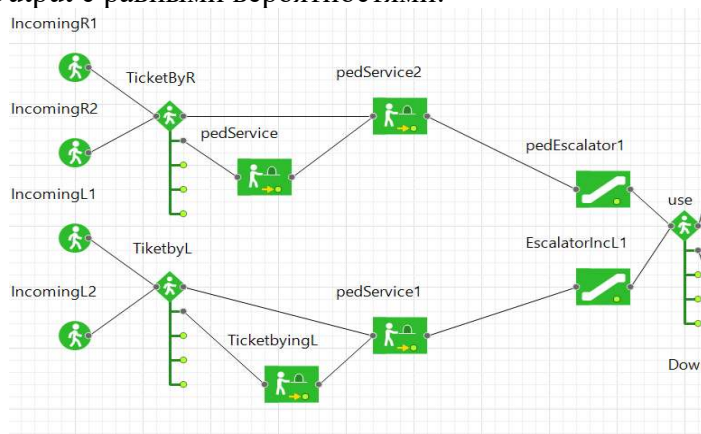


Рисунок 315. Добавление блока `ped SelectOutput`

Настройки выбора:

use - PedSelectOutput

Имя:

Использовать: ☒ Вероятности
☐ Условия
☐ Номер выхода

Козфф. предпочтения 1:

Козфф. предпочтения 2:

Рисунок 316. Настройка блока *ped SelectOutput*

Для каждого выбранного поезда включаем ветвление на 6 вагонов через применение 2 элементов выбора:

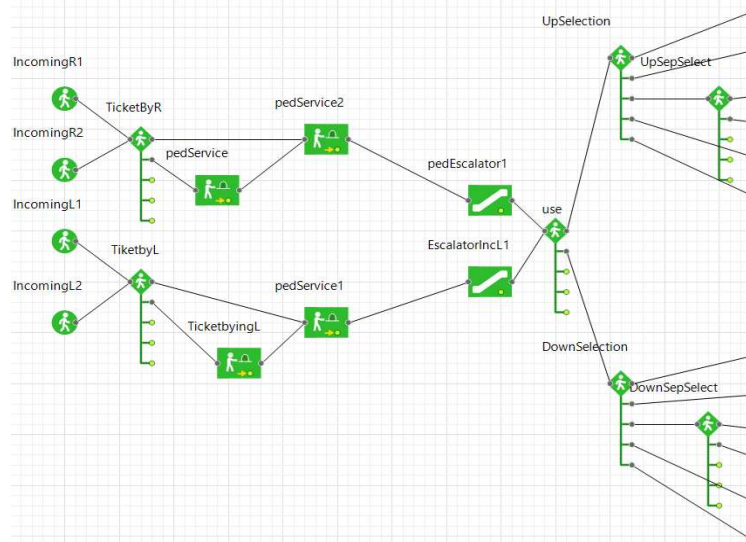


Рисунок 317. Добавление блоков *ped SelectOutput*

Для каждого вагона алгоритм одинаковый: пройти до линии ожидания (*ped go to*), встать в ожидание (*ped wait*), и по срабатыванию триггера заходить в вагон (*ped go to*, *sink*):



Рисунок 318. Блоки *pedGoTo* и *pedWait*

В настройках первых *ped go to* указываем необходимые целевые линии:

ToUpWait1 - PedGoTo

Имя:

☒ Отображать имя ☐ Исключить

Режим: ☒ Достичь цели
☐ Следовать по заданному пути

Цель: ☒ линия
☐ точка (x, y)
☐ область

Целевая линия:

Рисунок 319. Настройка блоков *pedGoTo*

Для *ped wait* настроим следующее состояние:

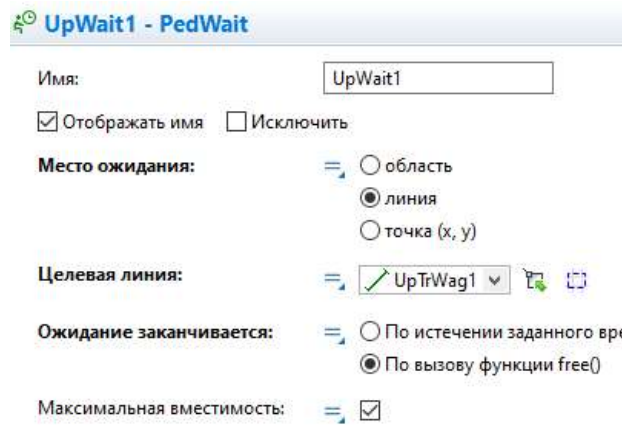


Рисунок 320. Настройка блока *pedWait*

Для *ped go to*, реализующего выход (заход в вагон) настраиваем целевую линию выхода:

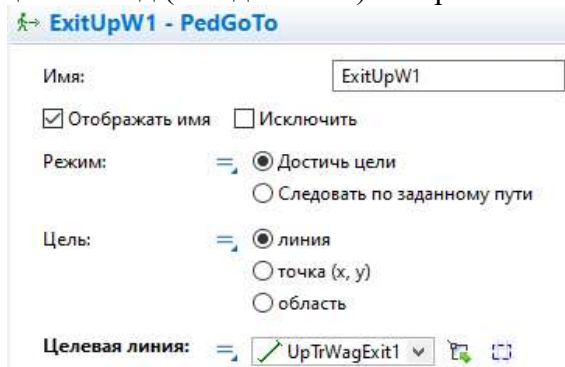


Рисунок 321. Настройка блоков *pedGoTo*

Завершаем всю систему блоком *sink* и получаем:

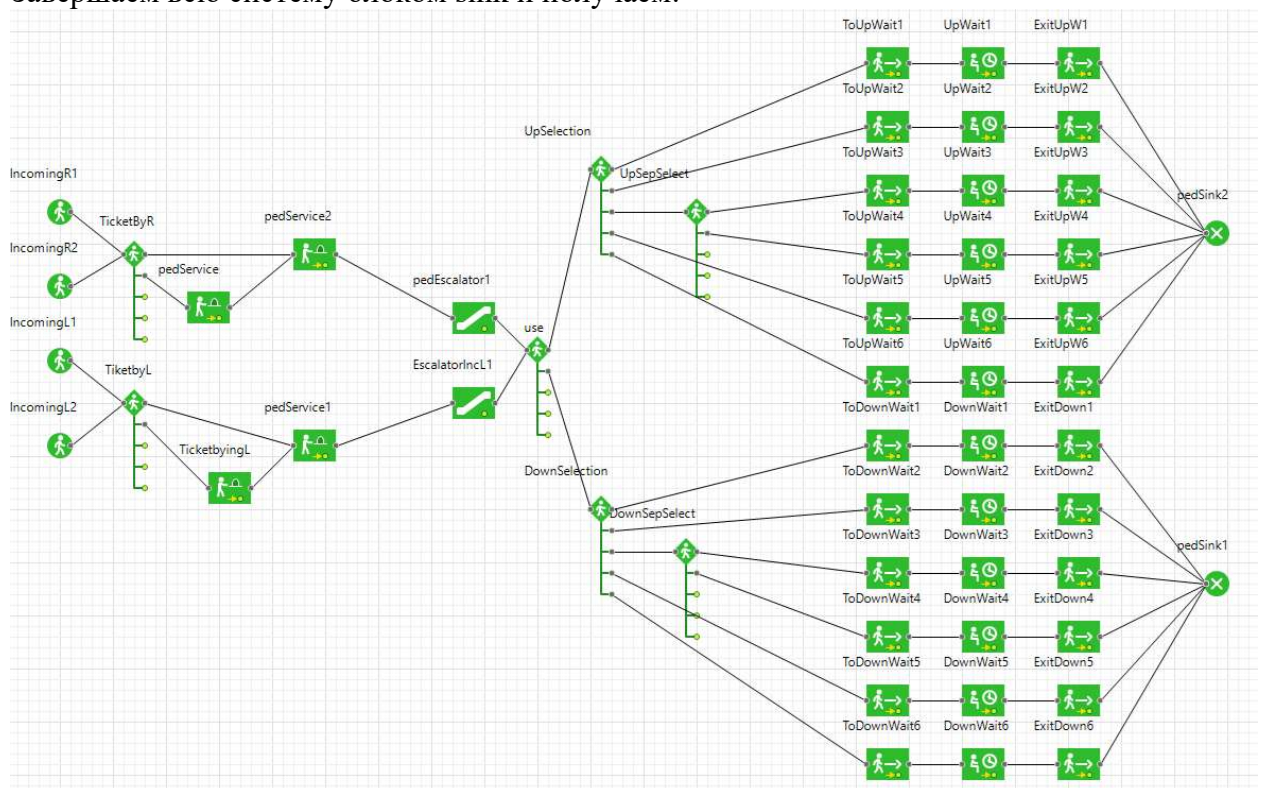


Рисунок 322. Логическая схема модели

Этап 4. Синхронизация пешеходной и Ж/д подсистем

Мы заметили, что для освобождения пешеходов для посадки в поезд в блоке *ped wait* требуется вызов функции *free()*. Добавим эту функцию в процесс работы Ж/д подсистемы: в блоках *Delay* пропишем в свойствах «Действий при входе»: #Название блока#.freeAll():

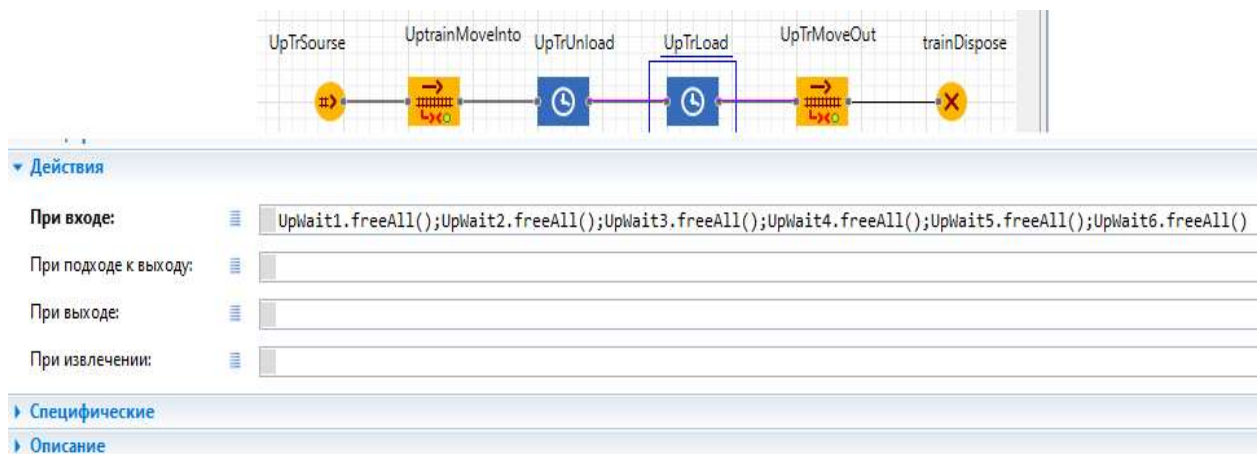


Рисунок 323. Настройка блоков Delay

Таким образом мы реализовали процесс «загрузки» пассажиров в поезд после определенного этапа простоя поезда на станции. С очевидностью можно сказать, что с приездом поезда мы должны вызвать функцию «выгрузки» пассажиров в модель. Добавим это в предыдущий блок Delay на этапе «при входе»:

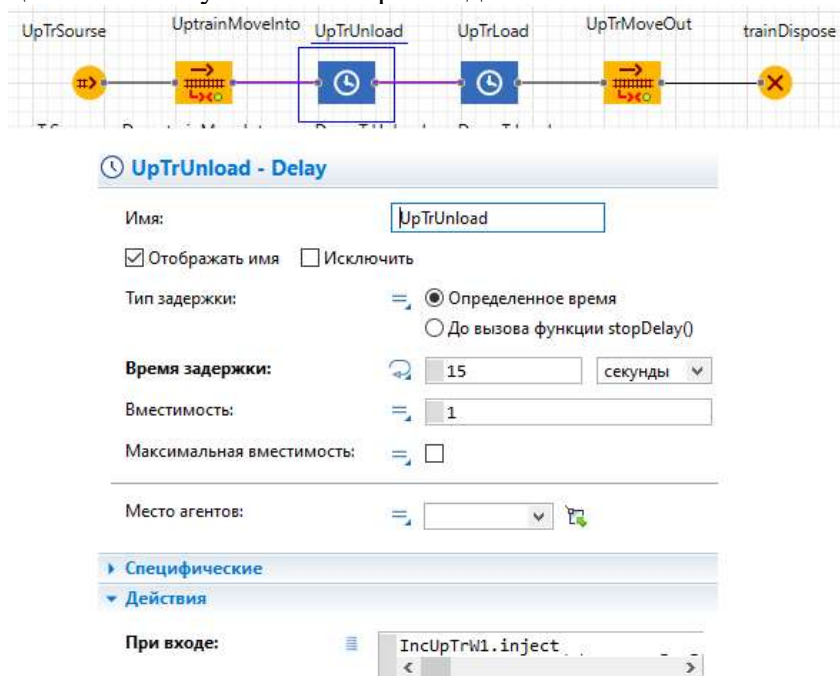


Рисунок 324. Настройка блоков Delay

Функцию #Название блока#.inject(N), где N – любое натуральное число (можно взять 10). Функция должна вызывать inject – «впрыск» пешеходов и целевых линий с каждого из 6 выходов вагона (то есть в теле действия надо прописать функцию для каждого выхода).

Этап 5. Реализация логической схемы пешеходной подсистемы высадки из поезда

Система выглядит следующим образом:

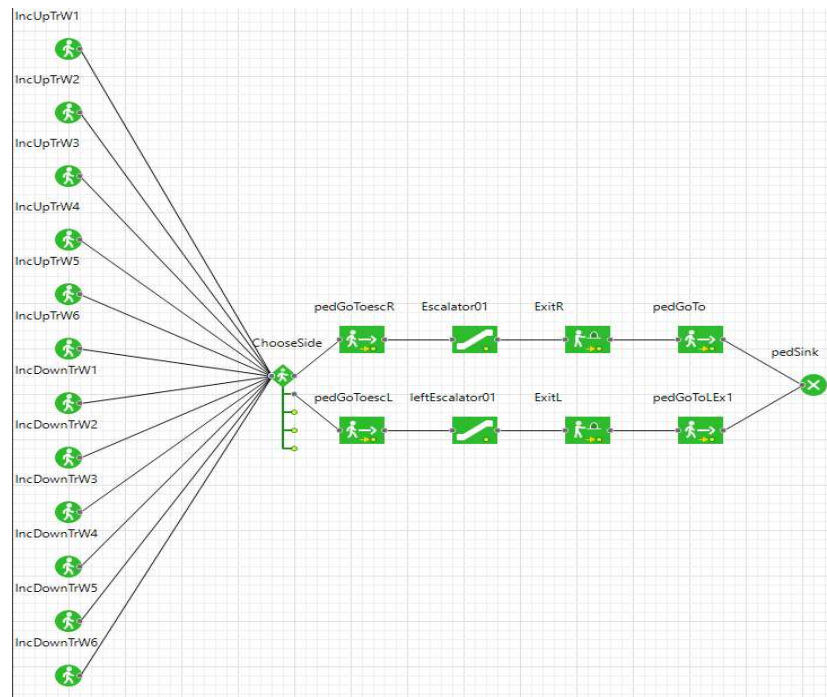


Рисунок 325. Логическая схема для высадки из поезда

Настройка блоков red source выглядит единообразно: привязка элемента разметки пространства, указание прибытия согласно функции inject():

IncUpTrW1 - PedSource

Имя:

☐ Отображать имя ☐ Исключить

Место появления: ☒ линия
☐ точка (x, y)
☐ область

Целевая линия:

Прибывают согласно:

Рисунок 326. Настройка блоков redSource

Вам уже должно быть понятно, что выход из любого вагона может вести к любому выходу со станции, поэтому все вагоны объединяют свой переход на выбор выхода в один равновероятный. Его логика одинакова и описывает простой алгоритм: дойти до зоны перед эскалатором, подняться на эскалаторе, пройти турникет, дойти до выхода со станции. Реализация этой несложной цепи остается в качестве самостоятельной работы.

Этап 6. Запуск модели

Запустите модель, пронаблюдайте за движением людей по станции.

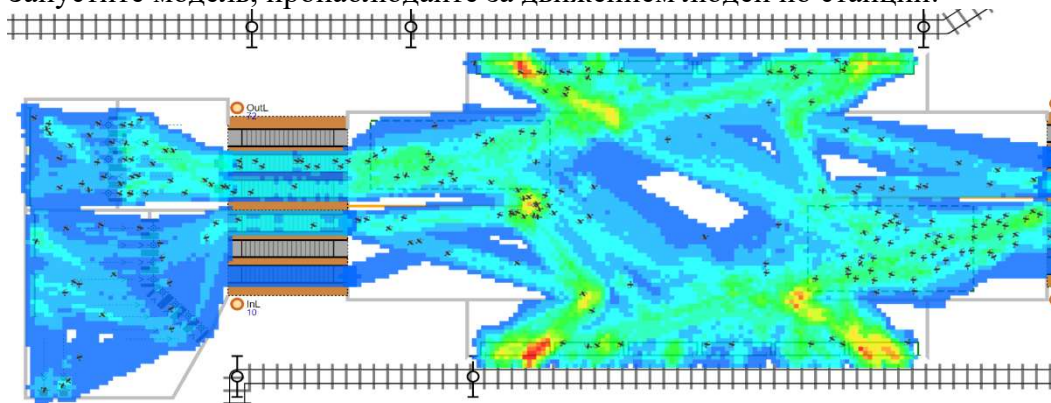


Рисунок 327. Анимация работы модели

Шаг 4. Программное управление эскалаторами

Внутри каждой группы эскалаторов можно управлять каждой отдельной дорожкой по отдельности. Отдельный эскалатор можно включить или выключить – тогда пешеходам можно им пользоваться как простой лестницей, либо можно «изолировать» – тогда эскалатор станет недоступным для пользования, однако если на нем в тот момент будут находиться пешеходы – они останутся там до снятия состояния изоляции. Воспользуемся возможностью включения/выключения эскалатора для решения задачи оптимизации формального энергопотребления. Будем считать, что нам важно, чтобы эскалаторы были загружены так, чтобы пешеходы не скапливались ни на платформе, ни на входе. При этом будем считать избыточным применение эскалаторов для перевозки менее чем 20 человек.

Этап 1. Введение в модель переменных

Для регулировки работы эскалаторов включим в модель 4 дополнительных переменных с панели «Агент»:

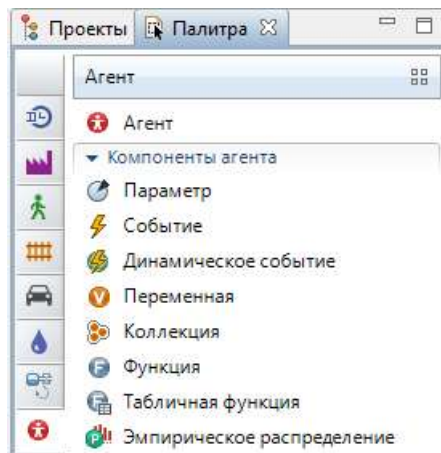


Рисунок 328. Палитра

Назовем их «InR» «OutR» «InL» «OutL» – они будут в себе носить данные по количеству пешеходов, переходящих к этапу пользования эскалатором – на вход или выход (In_, Out_) слева и справа (_L, _R) соответственно.

Этап 2. Управление переменными

Рассмотрим переменную «InL» – ответственную за количество пассажиров, находящихся на левой группе эскалаторов, и следующих вниз – на платформу. Логично, что она должна увеличиваться когда пассажир пройдет турникет, и уменьшаться когда он закончит движение на эскалаторе. Не стоит уравнивать события «на входе» эскалатора и «на выходе» турникетов – так как нам нужно управлять эскалаторами по потоку, поступающему к эскалатору, а не уже находящимся на нем.

В свойствах блока ped Service настроим действия при выходе с очереди: InL++.

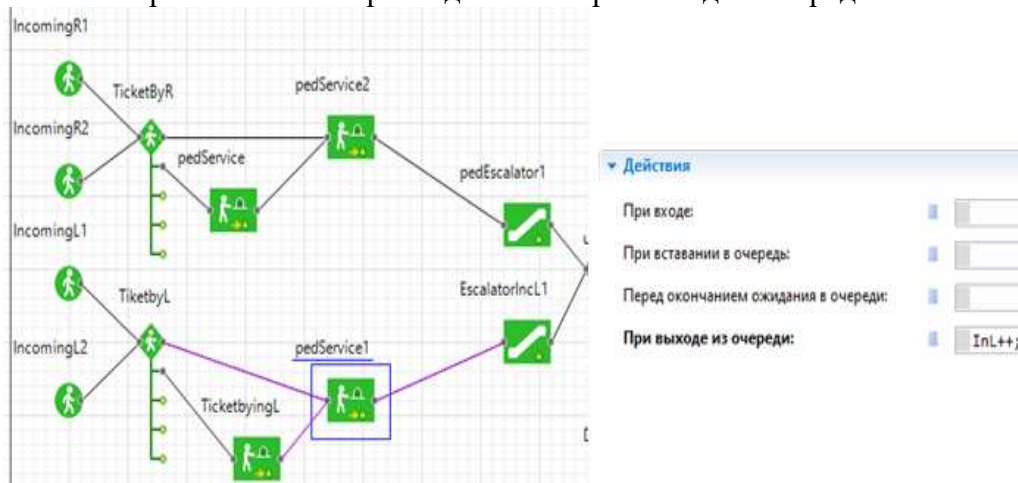


Рисунок 329. Настройка блока pedService

На выходе блока `pedEscalator`, относящегося к левому спуску настроим действие: `InL--;`;

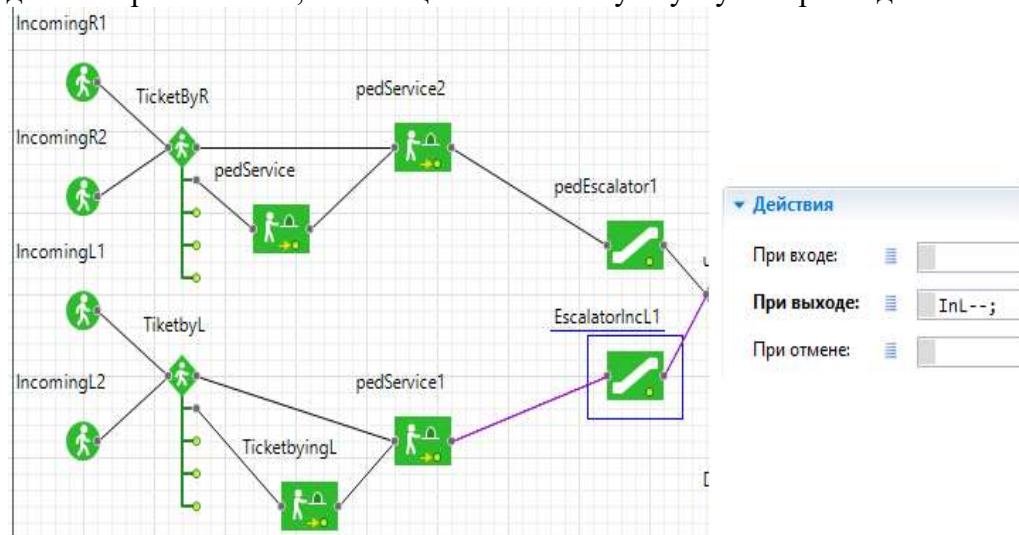


Рисунок 330. Настройка блока `pedEscalator`

Для остальных 3 переменных выполним аналогичные действия. Запустим модель и посмотрим на значения этих переменных:

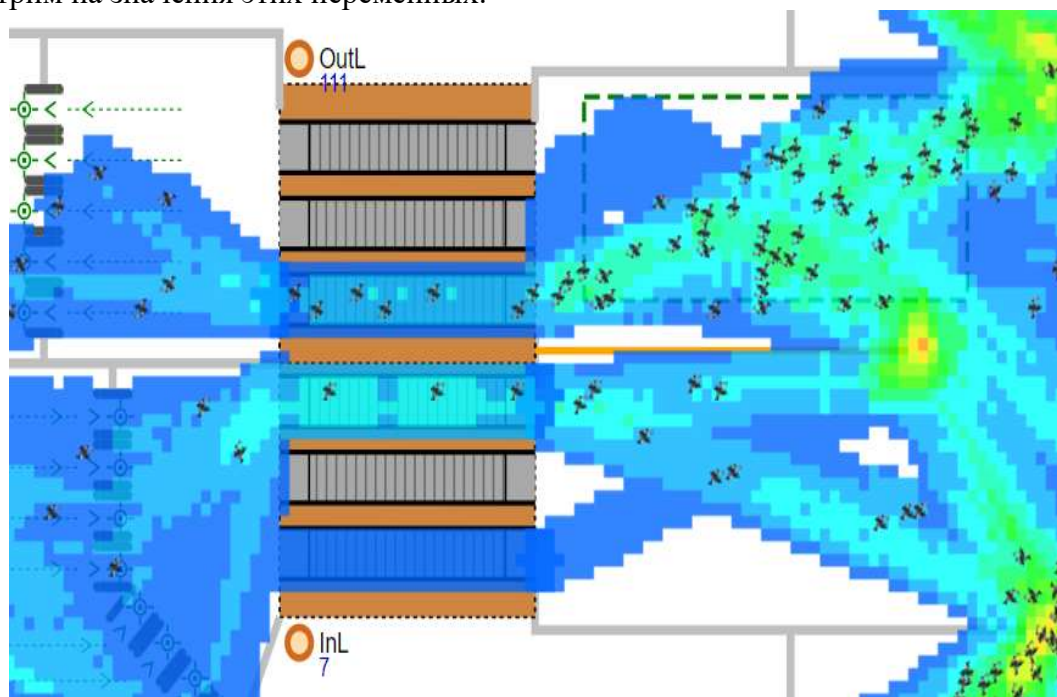


Рисунок 331. Анимация работы модели

Этап 3. Применение переменных, создание функции

Добавим 4 функции в модель. Они находятся в палитре «Агент». Функции будут управлять включением и выключением эскалаторов в зависимости от величины соответствующей переменной. В теле функции, управляющей левой группой спускающих эскалаторов пропишем:

function3 - Функция

Имя: ☒ Отображать имя ☐ Исключить

Видимость: ☒ да

☒ Действие (не возвращает ничего)
☐ Возвращает значение

Аргументы

Имя	Тип

Тело функции

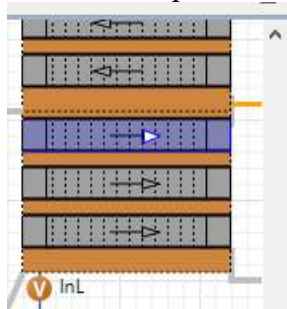
```

int x=InL;
if ((x>0))
{esl3.turnOn();
esl2.turnOff();
esl1.turnOff();
}
if ((x>20))
{esl3.turnOn();
esl2.turnOn();
esl1.turnOff();
}
if (x>40)
{esl3.turnOn();
esl2.turnOn();
esl1.turnOn();
}

```

Рисунок 332. Добавление функций

Здесь esl1-3 – названия дорожек эскалаторов выбранной группы:



esl3 - Эскалатор

Имя: ☐ Исключить

☒ Отображается на верхнем агенте ☐ Блокировать

Направление: ☐ Вверх
☒ Вниз

Описание

Рисунок 333. Настройка эскалаторов

Функция будет включать и выключать дорожки эскалаторов в зависимости от количества пешеходов на них – для 0-20 будет работать 1 дорожка, для 20-40 – будут работать 2, а для более чем 40 – все дорожки. Для эскалаторов, поднимающих пешеходов рекомендуется поставить границы шире – 0, 80, 160. Можно сделать вывод, что в нашей модели пешеходов с поезда поступает больше, чем с входа поверхности.

Функция требует запуска – включим это в модель. Будем запускать ее на выходе со всей группы турникетов: пропишем действие в свойствах блока:

▼ Действия	
При входе:	<input type="text"/>
При вставании в очередь:	<input type="text"/>
Перед окончанием ожидания в очереди:	<input type="text"/>
При выходе из очереди:	<input type="text" value="InL++;"/>
При начале обслуживания:	<input type="text"/>
При окончании обслуживания:	<input type="text"/>
При выходе:	<input type="text" value="function3();"/>
При отмене:	<input type="text"/>

Рисунок 334. Настройка турникетов

Этап 4. Запуск модели

Запустим модель и увидим, что для малого пассажиропотока (по 1000 человек в час) у нас будет работать только 1 эскалатор – значение переменной не будет превышать 20. Но если в настройках источника пешеходов задать интенсивность 4000 человек в час, можем получить в периоды экстремальной нагрузки значения переменной вплоть до 50, что хорошо будет реализовываться на 3 линейках эскалаторов:

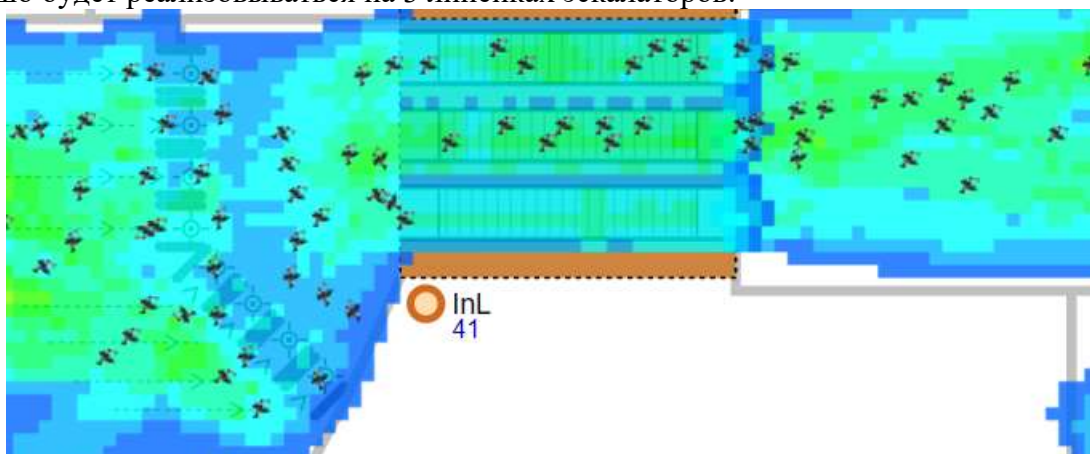


Рисунок 335. Анимация работы модели

Возможные дополнительные блоки для самостоятельной работы

Очевидно, модель имеет множество допущений и упрощений – так, мы не занимались настройкой визуализации поездов, реализовывали работу только 1 станции, не вели учет проходящих пассажиров.

Все эти расширения остаются для самостоятельной работы.