Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

А. С. Никитин, М. Ю. Чураков

Моделирование проезда нерегулируемых перекрёстков равнозначных дорог

Объектно-ориентированное программирование с явным выделением состояний

Проектная документация

Проект создан в рамках
«Движения за открытую проектную документацию»
http://is.ifmo.ru

Санкт-Петербург
2007

Оглавление

Введение

В существующих на данный момент правилах дорожного движения (ПДД) одним из наиболее сложных моментов является порядок разъезда на нерегулируемых перекрёстках равнозначных дорог. В материалах для подготовки к экзамену в ГИБДД обычно проводится детальный разбор лишь некоторых случаев. Однако, ситуации, которые возникают на практике, зачастую оказываются значительно сложнее, и требуют от водителя быстрого анализа дорожной обстановки и принятия решения. В таких случаях у водителя нет времени для того, чтобы перебрать в памяти все рассмотренные на уроках примеры и вспомнить, как необходимо действовать в конкретной ситуации. Для нормального управления автомобилем в сознании водителя должна быть чёткая схема действий для предотвращения аварийной ситуации. Задача обучения состоит как раз в том, чтобы сформировать в сознании учащегося ясный алгоритм принятия решения в любой дорожной обстановке и довести навыки управления транспортным средством до автоматизма. Основным правилом разъезда является правило «правой руки», предписывающее уступить дорогу, если есть помеха справа, однако на практике возможна неверная трактовка этого правила, которая приведет к аварии.

Данный проект разработан с целью наглядной демонстрации правил проезда нерегулируемых перекрёстков равнозначных дорог. Пользователю достаточно лишь задать расположение автомобилей и направления их движения, чтобы увидеть, как в соответствии с ПДД должны разъезжаться водители. Автоматный подход, применяемый авторами, позволяет эффективно реализовать механизм принятия решений по выбору траектории и порядку начала движений.

Проект выполнен на языке программирования *Java* в среде *Eclipse* с помощью инструментального средства *UniMod*.

1. Постановка задачи

Авторами ставилась задача моделирования проезда нерегулируемых перекрёстков равнозначных дорог с учётом всех имеющихся правил, представленных в соответствующих разделах ПДД. Кроме того, требовалась реалистичность визуальной составляющей – то есть, автомобили должны двигаться по естественным траекториям, не задевая друг друга.

Как и в любой другой модели, в данном проекте используются некоторые допущения:

- рассматривается только случай нерегулируемого перекрёстка равнозначных дорог;
- дороги имеют по одной полосе для движения в каждом направлении;
- все автомобили движутся с одинаковой скоростью;
- разворот по малой траектории (без выезда на середину перекрёстка) невозможен.

1.1. Правила разъезда

Приведём выдержки из ПДД, которые реализуются в данном проекте:

> 13.11. На перекрестке равнозначных дорог водитель безрельсового транспортного средства обязан уступить дорогу транспортным средствам, приближающимся справа.
>
> 13.12. При повороте налево или развороте водитель безрельсового транспортного средства обязан уступить дорогу транспортным средствам, движущимся по равнозначной дороге со встречного направления прямо или направо.

Существуют ситуации, когда все участники движения, формально действуя по правилам, должны оставаться на месте, потому что для каждого из них присутствует помеха в виде другого автомобиля, имеющего преимущество. В этом случае водители транспортных средств должны договориться между собой о том, кто первым проедет перекрёсток.

1.1.1. Пример разъезда на перекрёстке

Рассмотрим случай, изображённый на рис.1. Автомобили, приближающиеся снизу и справа, движутся в прямом направлении, а сверху и слева – налево и направо, соответственно. Порядок разъезда в этом случае определяется однозначно – лишь зелёная машина, находящаяся слева, не имеет помех, поэтому первой проезжает перекрёсток. Оранжевая машина не движется, так как зелёная для неё является помехой справа. Жёлтая машина не может повернуть налево, так как пропускает оранжевую, движущуюся со встречного направления. Синяя машина аналогично дожидается, пока исчезнет помеха справа.

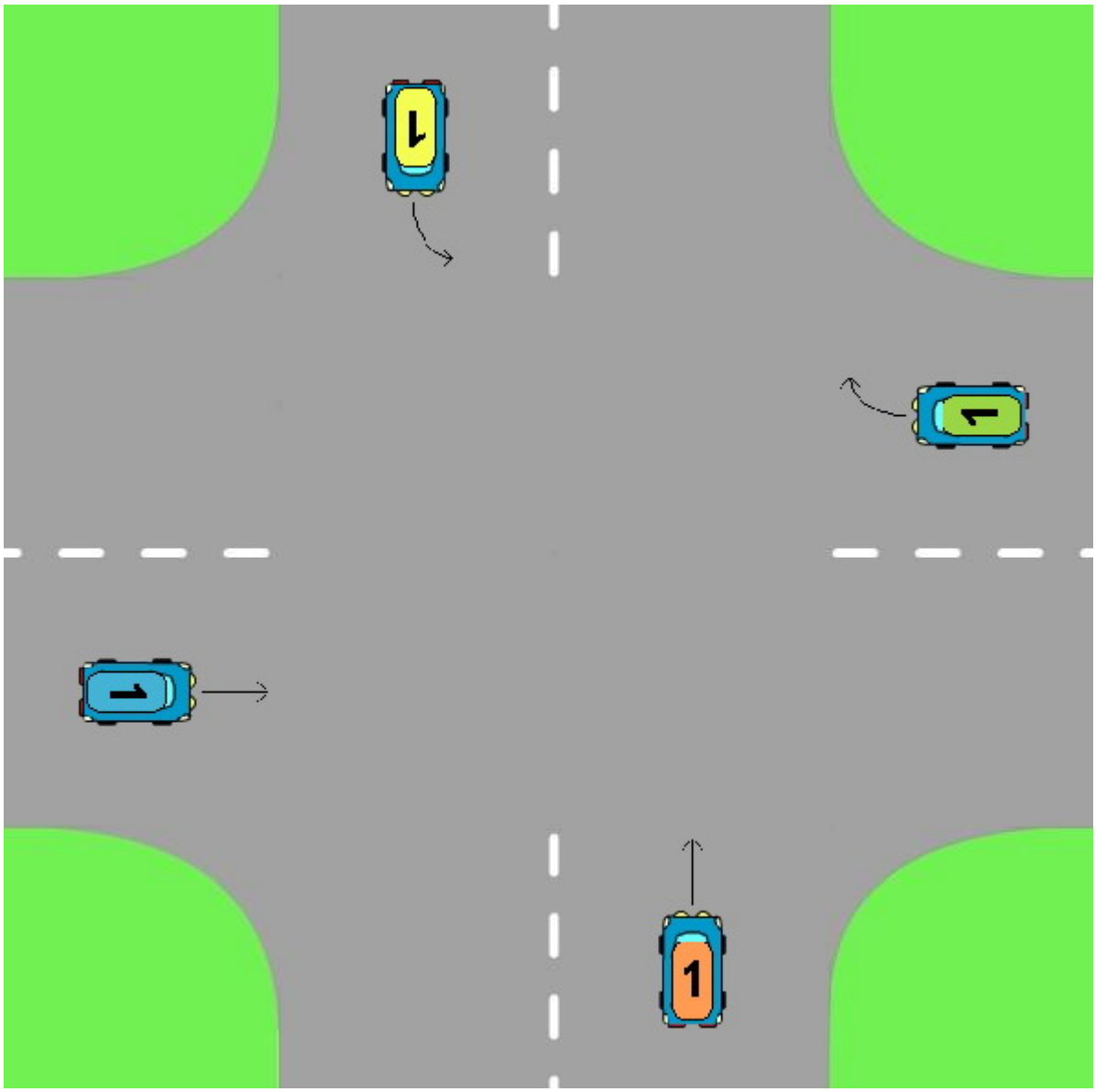

Рис. 1. Пример разъезда: направления движений

Одновременно с тем, как зелёная машина проезжает перекрёсток, жёлтая выезжает на середину перекрёстка, чтобы, пропустив оранжевую, закончить манёвр и покинуть перекрёсток. Далее для оранжевой машины исчезает помеха справа, и она может начать движение (рис. 2).

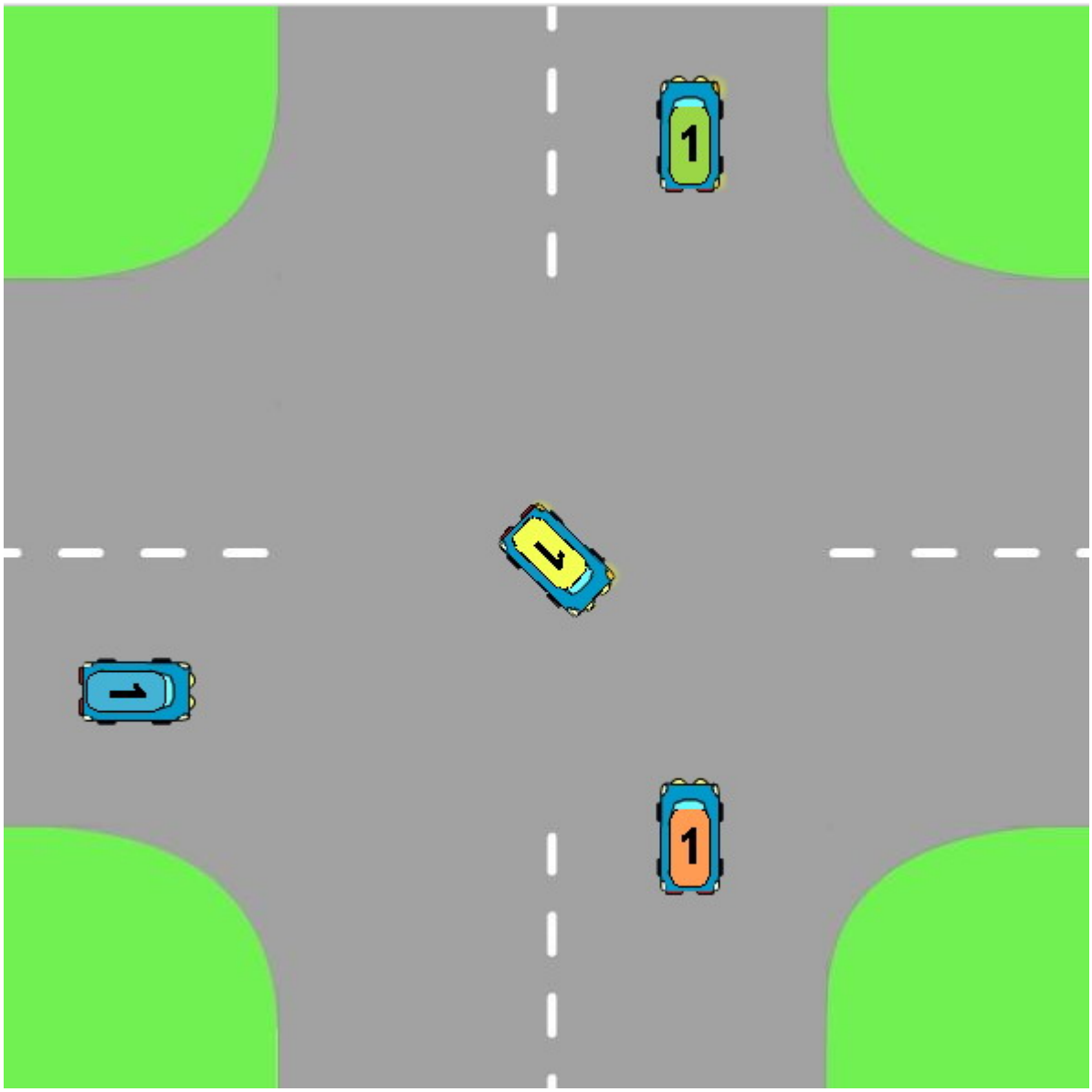

Рис. 2. Пример разъезда: зелёная машина поворачивает направо,
жёлтая – выезжает на середину перекрёстка, оранжевая – начинает двигаться

После того как оранжевая машина больше не создаёт помех, жёлтая и оранжевая машины могут продолжить движение. Но их траектории пересекаются, поэтому одновременно двигаться они не могут. В этом случае жёлтая машина в соответствии с пунктом 13.11 ПДД должна уступить дорогу синей, так как для неё она является приближающейся справа (рис. 3).

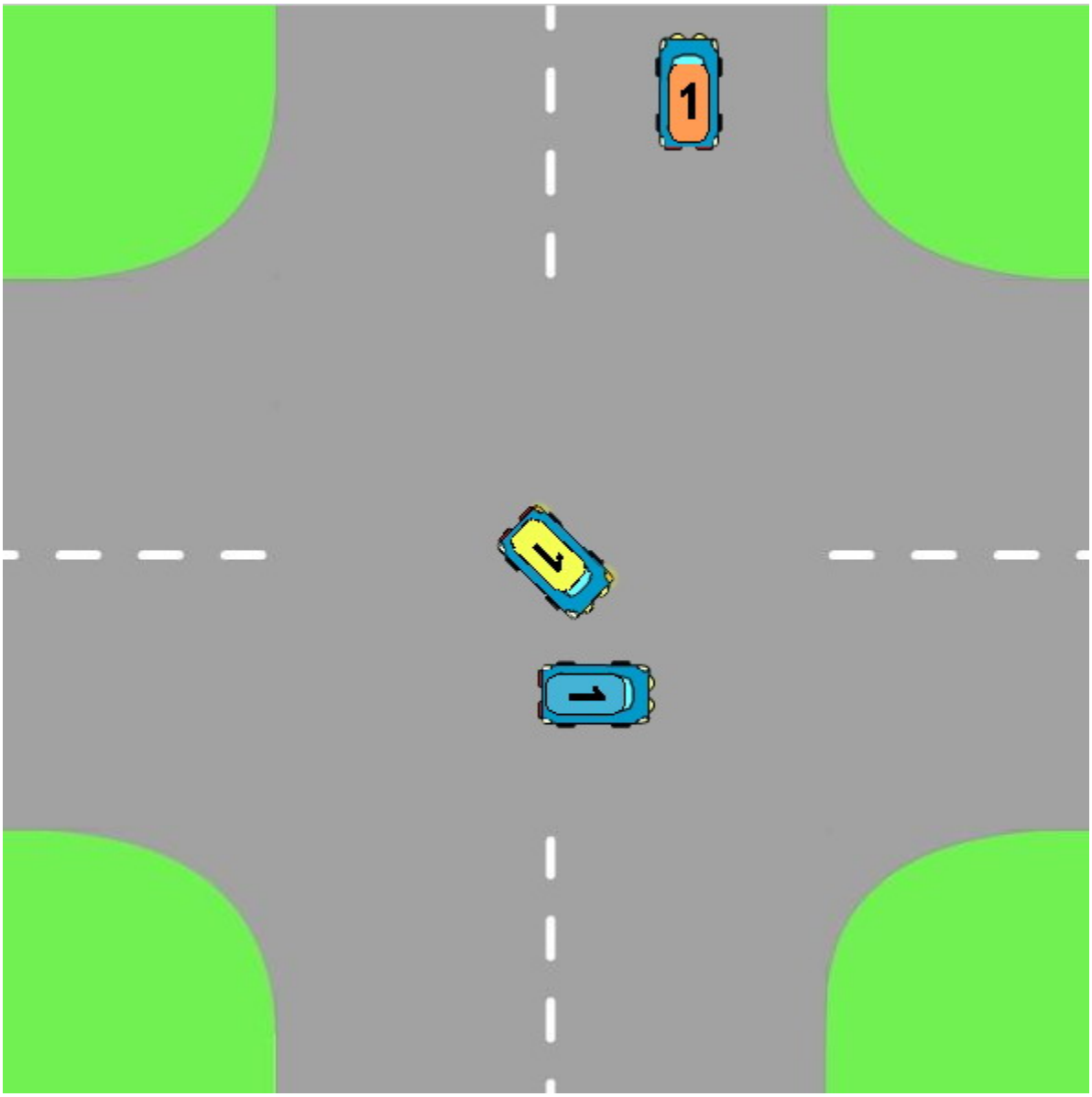

Рис. 3. Пример разъезда: жёлтая машина пропускает синюю

После этого жёлтая машина завершает поворот налево, не имея никаких помех, потому что все остальные участники движения уже успели покинуть перекрёсток.

1.2. Внешний вид программы и пользовательский интерфейс

После запуска программы (рис. 4) необходимо задать конфигурацию перекрёстка, то есть задать направления движения машин и механизмы их управления – один из трёх автоматов:

- автомат, реализующий поведение водителя, соответствующее ПДД;
- автомат, подчиняющийся ПДД, но не включающий сигналы поворота;
- автомат, который не соблюдает ПДД.

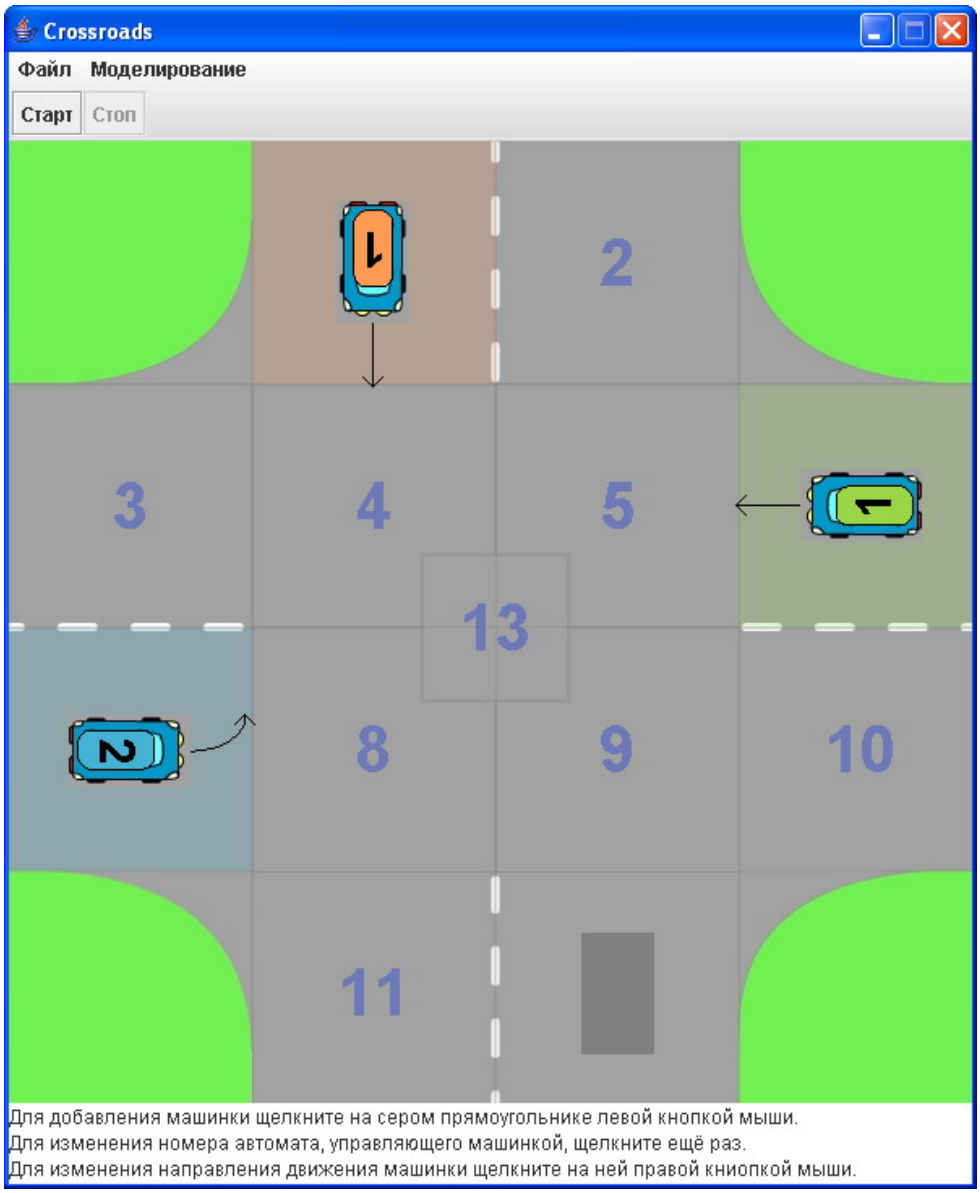

Рис. 4. Окно приложения

Для того чтобы установить машину, движущуюся с одного из четырёх направлений, достаточно щёлкнуть на сером прямоугольнике левой кнопкой мыши. Для смены автомата – нужно щёлкнуть ещё раз. Для изменения направления движения нужно щёлкнуть правой кнопкой мыши.

Также можно указать такие дополнительные параметры отображения, как траектории и зоны, которые подсвечиваются цветом, соответствующим машине, которая её занимает.

Далее всё управление осуществляется набором кнопок *Старт/Пауза/Продолжить* и *Стоп*.

После завершения моделирования можно задать другой перекрёсток, чтобы посмотреть, как разъедутся автомобили в другом случае. При этом можно сохранять и загружать конфигурации перекрёстков из файла.

2. Проектирование

2.1. Диаграмма классов

В проекте используются две *Unimod*-модели. Одна для управления интерфейсом (`core.unimod`), а другая для принятия решений о проезде перекрёстка (`crossroad.unimod`). Для их связи используются дополнительные классы (рис. 5), написанные на языке *Java*, которые обеспечивают всю низкоуровневую работу с данными, отвечающими за внутреннее представленит информации о перекрёстке и автомобилях.

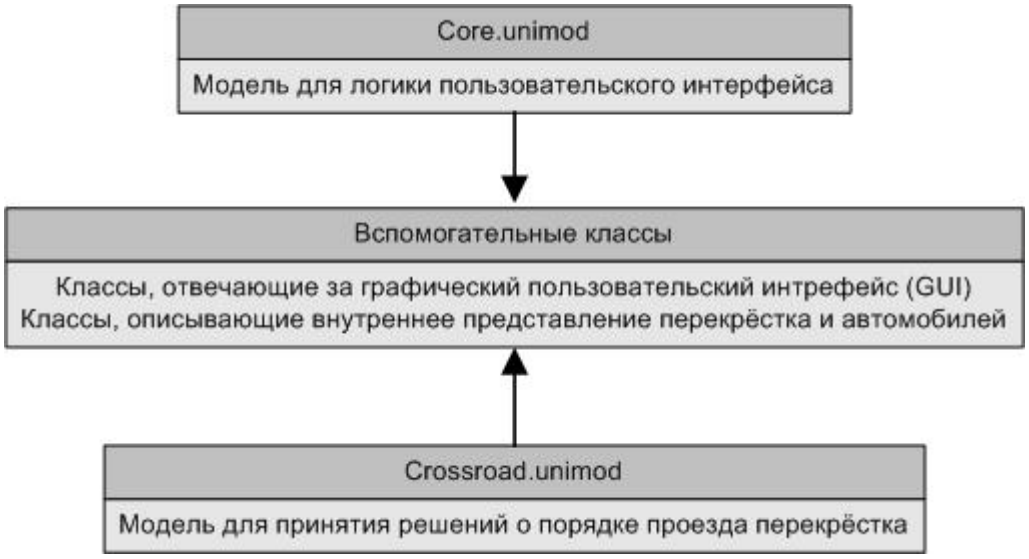

Рис.5. Общая схема классов

При запуске программа стартует из автомата `Main`, находящегося в `core.unimod`. Автоматы же из `crossroad.unimod` предварительно скомпилированы и динамически подключаются в процессе работы приложения.

2.2. Core.unimod

Как уже было сказано, в проекте используются две *Unimod*-модели. Одна для принятия решений о проезде перекрёстка (`crossroad.unimod`), а другая для управления интерфейсом (`core.unimod`). Остановимся на последней.

2.2.1. Диаграмма связей

Диаграмма связей системы управления интерфейсом (`core.unimod`). изображена на рис. 6.

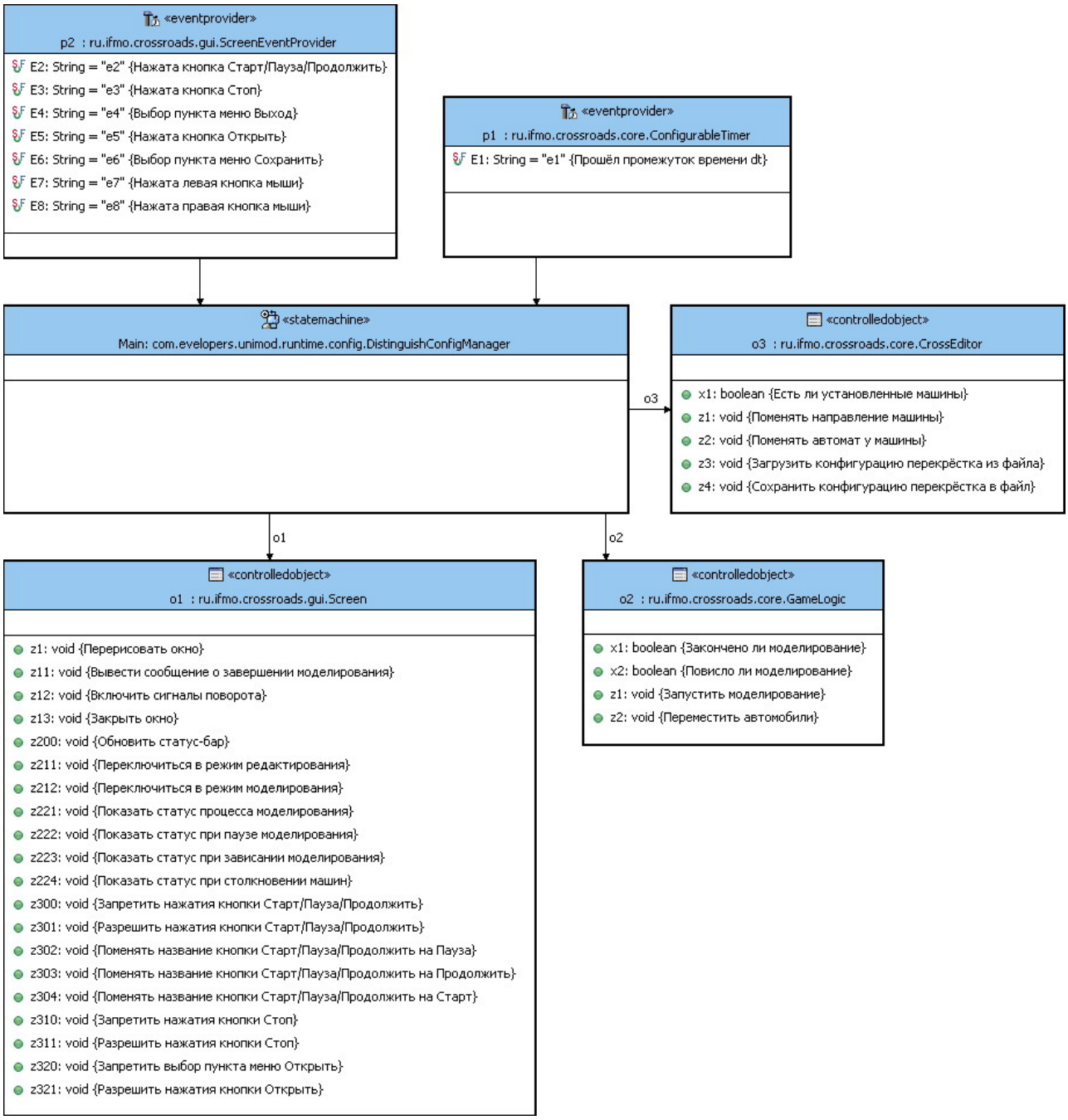

Рис. 6. Диаграмма связей `core.unimod`

Поставщики событий

ConfigurableTimer

Этот поставщик событий во время выполнения программы генерирует событие *E1* через определенный, заранее установленный промежуток времени `dt`. Является расширением стандартного поставщика событий `Timer`, который посылает событие лишь раз в секунду.

Событие	Описание
E1	Прошёл промежуток времени `dt`

ScreenEventProvider

Передаёт воздействия пользователя (нажатие кнопок и выбор пунктов меню) автомату `Main`.

Событие	Описание
E2	Нажата кнопка *Старт/Пауза/Продолжить*
E3	Нажата кнопка *Стоп*
E4	Выбор пункта меню *Выход*
E5	Нажата кнопка *Открыть*
E6	Выбор пункта меню *Сохранить*
E7	Нажата левая кнопка мыши
E8	Нажата правая кнопка мыши

Автомат Main

Диаграмма состояний автомата `Main` изображена на рис. 7.

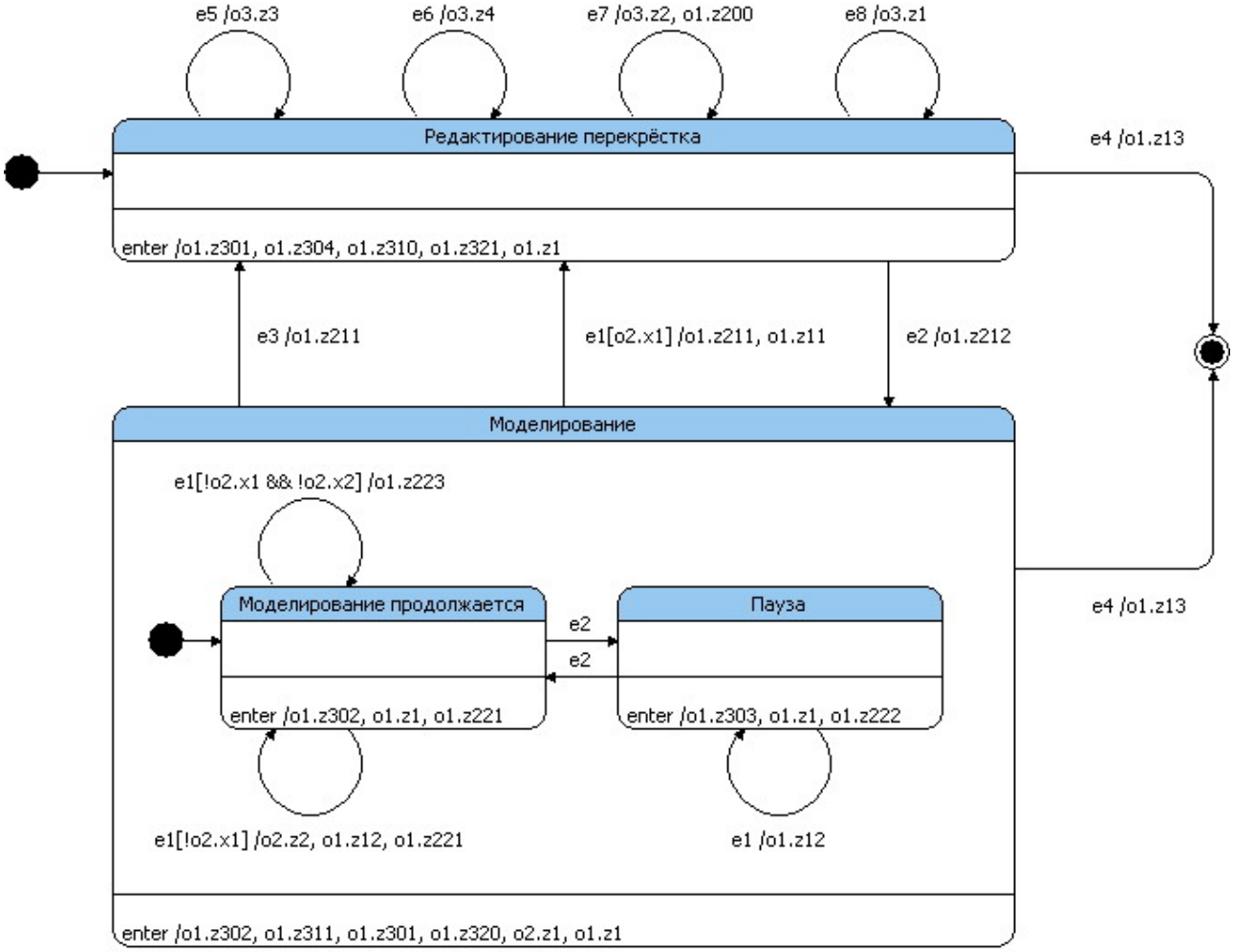

Рис. 7. Автомат `Main`

Управляет графическим интерфейсом пользователя, обеспечивая реагирование на нажатия кнопок и выборы пунктов меню. Вложенный автомат реализует главный цикл моделирования.

Объекты управления

Screen

Содержит входные и выходные воздействия, связанные с внешним видом окна и его перерисовкой.

Выходное воздействие	Описание
Z1	Перерисовать окно
Z11	Вывести сообщение о завершении моделирования
Z12	Включить сигналы поворота
Z13	Закрыть окно
Z200	Обновить статус-бар
Z211	Переключиться в режим редактирования
Z212	Переключиться в режим моделирования
Z221	Показать статус процесса моделирования
Z222	Показать статус при паузе моделирования
Z223	Показать статус при зависании моделирования
Z224	Показать статус при столкновении машин
Z300	Запретить нажатия кнопки *Старт/Пауза/Продолжить*
Z301	Разрешить нажатия кнопки *Старт/Пауза/Продолжить*
Z302	Поменять название кнопки *Старт/Пауза/Продолжить* на *Пауза*
Z303	Поменять название кнопки *Старт/Пауза/Продолжить* на *Продолжить*
Z304	Поменять название кнопки *Старт/Пауза/Продолжить* на *Старт*
Z310	Запретить нажатия кнопки *Стоп*
Z311	Разрешить нажатия кнопки *Стоп*
Z320	Запретить выбор пункта меню *Открыть*
Z321	Разрешить нажатия кнопки *Открыть*

GameLogic

Содержит входные и выходные воздействия, связанные с процессом моделирования.

Входное воздействие	Описание
X1	Закончено ли моделирование
X2	Есть ли прогресс в моделирование (не повисло ли)

Выходное воздействие	Описание
Z1	Запустить моделирование
Z2	Переместить автомобили

CrossEdit

Содержит входные и выходные воздействия, связанные с процессом редактирования перекрёстка.

Входное воздействие	Описание
X1	Есть ли установленные машины

Выходное воздействие	Описание
Z1	Поменять направление машины
Z2	Поменять автомат у машины
Z3	Загрузить конфигурацию перекрёстка из файла
Z4	Сохранить конфигурацию перекрёстка в файл

2.3. Crossroad.unimod

2.3.1. Схема взаимодействия автоматов

После того, как пользователь нажал кнопку *Старт,* создаётся экземпляр класса `Cross`, который отвечает за моделирование перемещения машин. Объект `Cross` при инициализации получает структуру данных, в которой указана конфигурация перекрестка, то есть расположение машин, их направления движения и номера автоматов, управляющих ими. По этой информации `Cross` загружает `xml`-модели автоматов машин, и связывает каждый автомат с экземпляром класса `CarCO`, инкапсулирующего отдельную машину.

Таким образом, за исключением двух сообщений, получаемых от объекта `Cross` при начале моделирования, каждый автомат взаимодействует только со своим собственным экземпляром класса `CarCO`, который является для него одновременно и поставщиком событий и объектом управления.

Объект `Cross` служит диспетчером событий для объектов `CarCO`: каждое сообщение, сгенерированное каким-либо объектом `CarCO` направляется в объект Cross, где оно транслируется другим объектам `CarCO`, а те, в свою очередь, передают его своим автоматам (рис. 8).

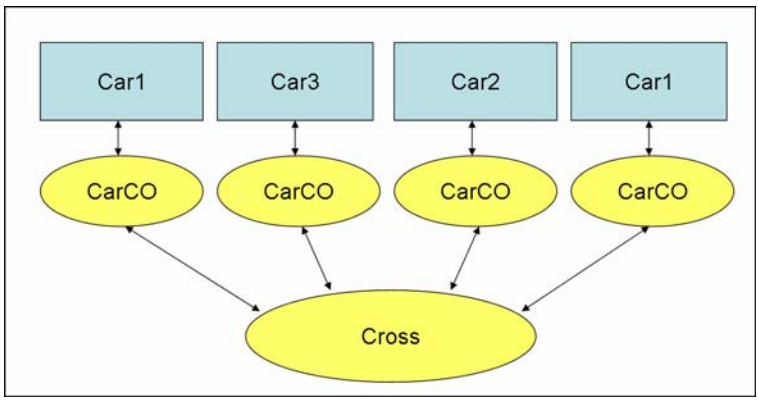

Рис. 8. Схема взаимодействия автоматов

2.3.2. Модель перекрёстка

Территория перекрестка разбивается на 13 зон, как показано на рис. 9.

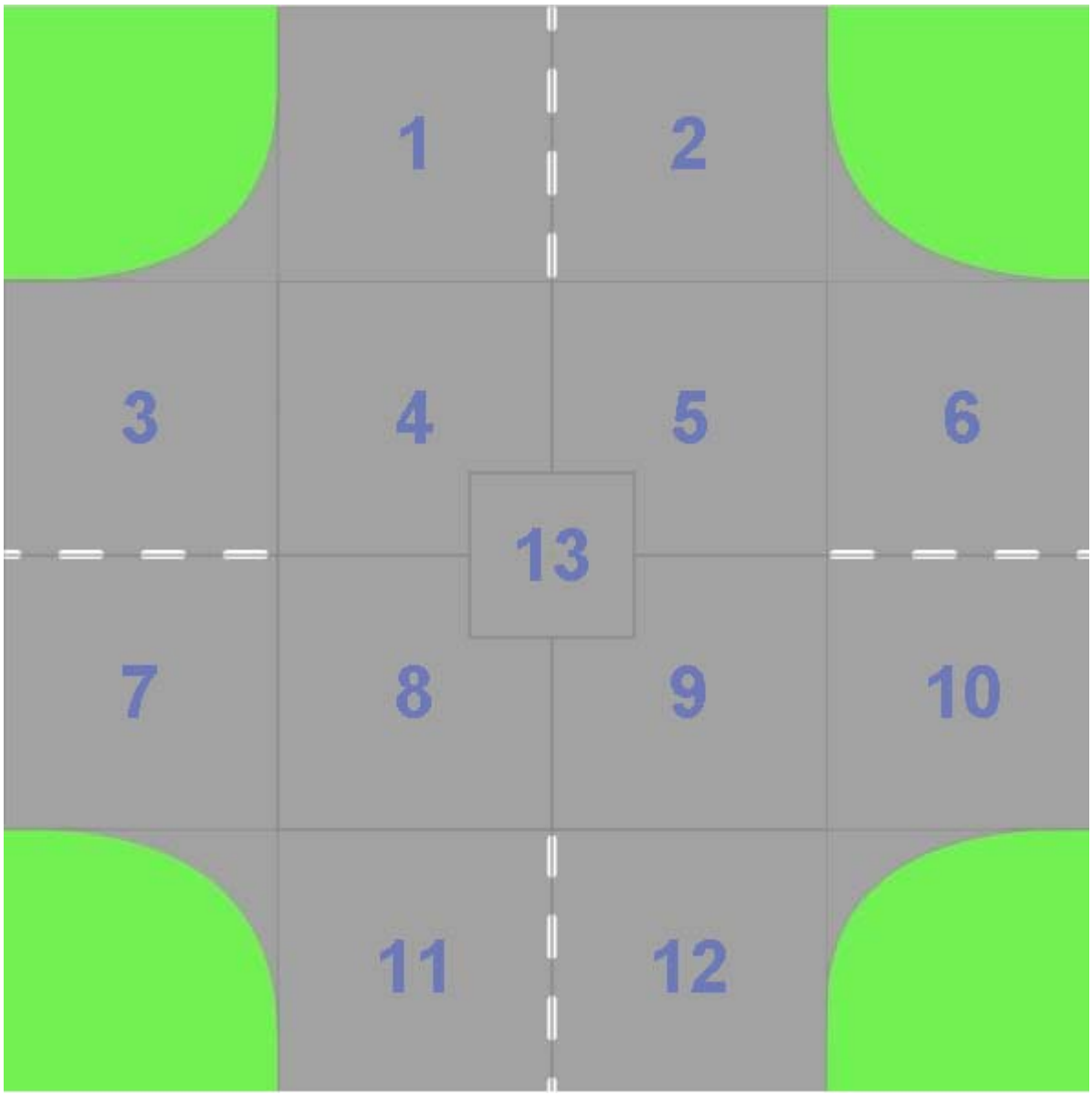

Рис. 9. Разбиение перекрёстка на зоны

Нумерация зон ведется относительно каждой конкретной машинки, то есть для каждой машины встречная машинка будет появляться в зоне 1, левая – в зоне 7, правая – в зоне 6 и т.д.

Машина может двигаться по одной из восьми возможных траекторий: одна траектория поворота направо, одна – прямо (рис. 10), три – налево (рис. 11) и три – на разворот (рис. 12). (авторами рассматривается случай, когда разворот по малой траектории, не выходящей за пределы зон 8 и 9, невозможен.) Траектория выбирается автоматом в начале моделирования и не может быть изменена после того, как машина начала движение.

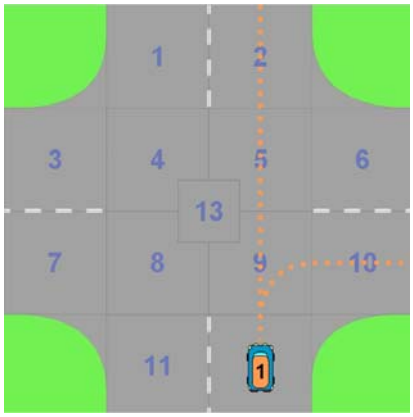

Рис. 10.
Траектории движения прямо и направо

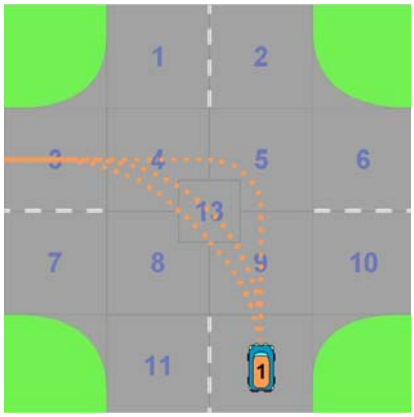

Рис. 11.
Траектории движения налево

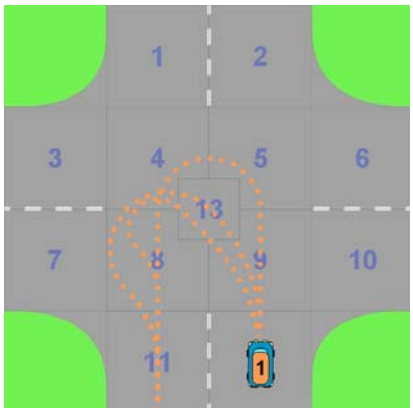

Рис. 12.
Траектории разворота

Направление движения каждой машины задается при редактировании конфигурации. Если выбрано направление налево или на разворот, то конкретная траектория движения (одна из трёх) выбирается автоматом в начале моделирования.

Каждая траектория проходит по нескольким зонам перекрестка. Во время движения объект `CarCO` машины отслеживает номер зоны, в которой находится машина. При этом предполагается, что машина находится в той зоне, в которую попадает точка, являющаяся центром машины. При смене зоны объект `CarCO` генерирует сообщение о покидании зоны, которое через объект `Cross` транслируется остальным машинам.

Кроме сообщений о покидании зон другими машинами, автомат получает сообщения, когда его машина пересекает одну из разделительных полос.

Моделирование начинается с того, что объект `Cross` рассылает два сообщения всем машинам:

1. «Включить указатели поворота».
   При получении этого сообщения автомат должен включить необходимые указатели поворота для того, чтобы другие машины могли правильно оценить дорожную обстановку и выбрать тактику действий.
2. «Начать движение».
   При получении этого сообщения автомат может выбрать траекторию движения (если машина двигается налево или на разворот), и, если необходимо, начать движение.

После выбора траектории движения автомат может совершать только два выходных воздействия: «ехать» и «ждать». Для принятия решения автомат использует входные переменные, которые характеризуют окружающую обстановку: расположение машин на перекрестке и их указатели поворота.

2.3.3. Диаграмма связей

На рис. 13 изображена диаграмма связей с тремя однотипными автоматами для управления машиной.

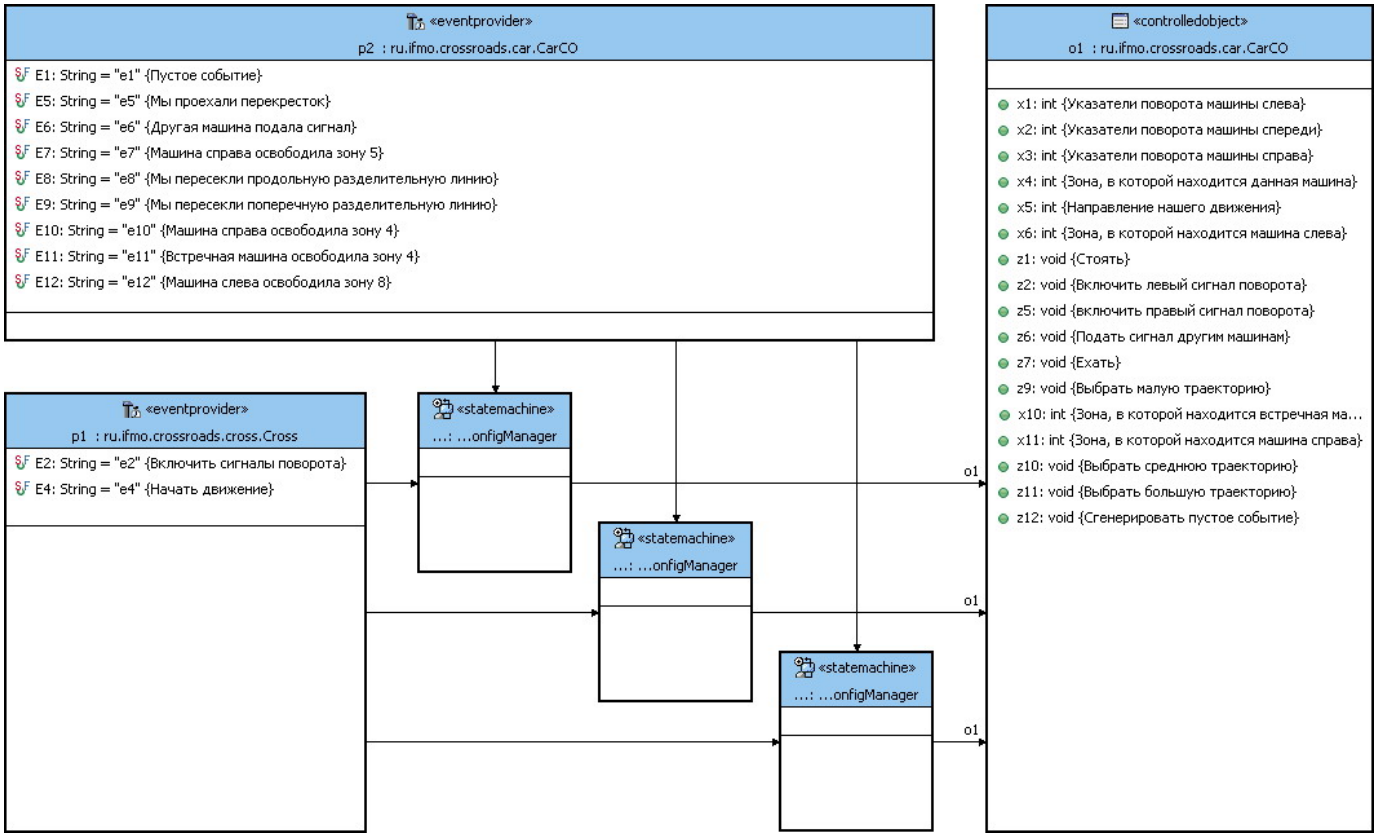

Рис. 13. Диаграмма связей `crossroad.unimod`

Поставщики событий

CarCO

Объект `CarCO` инкапсулирует отдельную машину и отвечает за моделирование перемещения вдоль траектории. Каждая машина может находиться в одном из четырёх состояний:

- MOVING – машина едет;
- WAITING – машина стоит на месте;
- CRASHED – машина столкнулась с другой машиной;
- FINISHED – машина проехала перекресток.

Объект `CarCO` является основным поставщиком событий для автомата.

Событие	Описание
E1	Событие по умолчанию
E5	Данная машина проехала перекрёсток
E6	Другая машина подала сигнал
E7	Машина справа покинула зону 5
E8	Данная машина пересекла продольную разделительную линию

E9	Данная машина пересекла поперечную разделительную линию
E10	Машина справа покинула зону 4
E11	Встречная машина покинула зону 4
E12	Машина слева покинула зону 4

Объект `CarCO` отвечает за перемещение машин вдоль выбранной траектории и отслеживание пересекаемых зон. Как только машина покидает какую-либо зону, объект `CarCO` отправляет соответствующее сообщение объекту `Cross` для трансляции остальным машинам.

Сообщения, получаемые автоматом от объекта `CarCO` можно разделить на два типа:

1. Собственные сообщения.
   К ним относятся три сообщения: два о пересечении разделительной полосы (продольной и поперечной) и сообщение о завершении движения.
2. Сообщения от других машин.
   Эти сообщения генерируются другими объектами `CarCO` и транслируются через объект `Cross`. К ним относятся все сообщения об освобождении зон и специальное сообщение «Кто-то махнул рукой», которое служит для разрешения ситуации, когда ни одна из машин не может начать движения.

Приведем пример трансляции сообщений. Допустим, машина слева покинула зону 8 (рис. 14). Относительно машины слева зона 8 является зоной 9, а зона 9 – зоной 5. Сообщения будут передаваться следующим образом:

1. Объект `CarCO` машины слева фиксирует, что машина перешла из зоны 9 в зону 5, генерирует сообщение «Я покинула зону 9» и передаёт его объекту `Cross`
2. Объект `Cross` переводит это сообщение в следующее: «Машина слева покинула зону 8»
3. Объект `Cross` отправляет сообщение «Машина справа покинула зону 5» объекту `CarCO` машины сверху
4. Объект `CarCO` машины сверху генерирует сообщение *E7* и отправляет его своему автомату
5. Объект `Cross` отправляет сообщение «Встречная машина покинула зону 4» объекту `CarCO` машины справа
6. Объект `CarCO` машины справа генерирует сообщение *E11* и отправляет его своему автомату

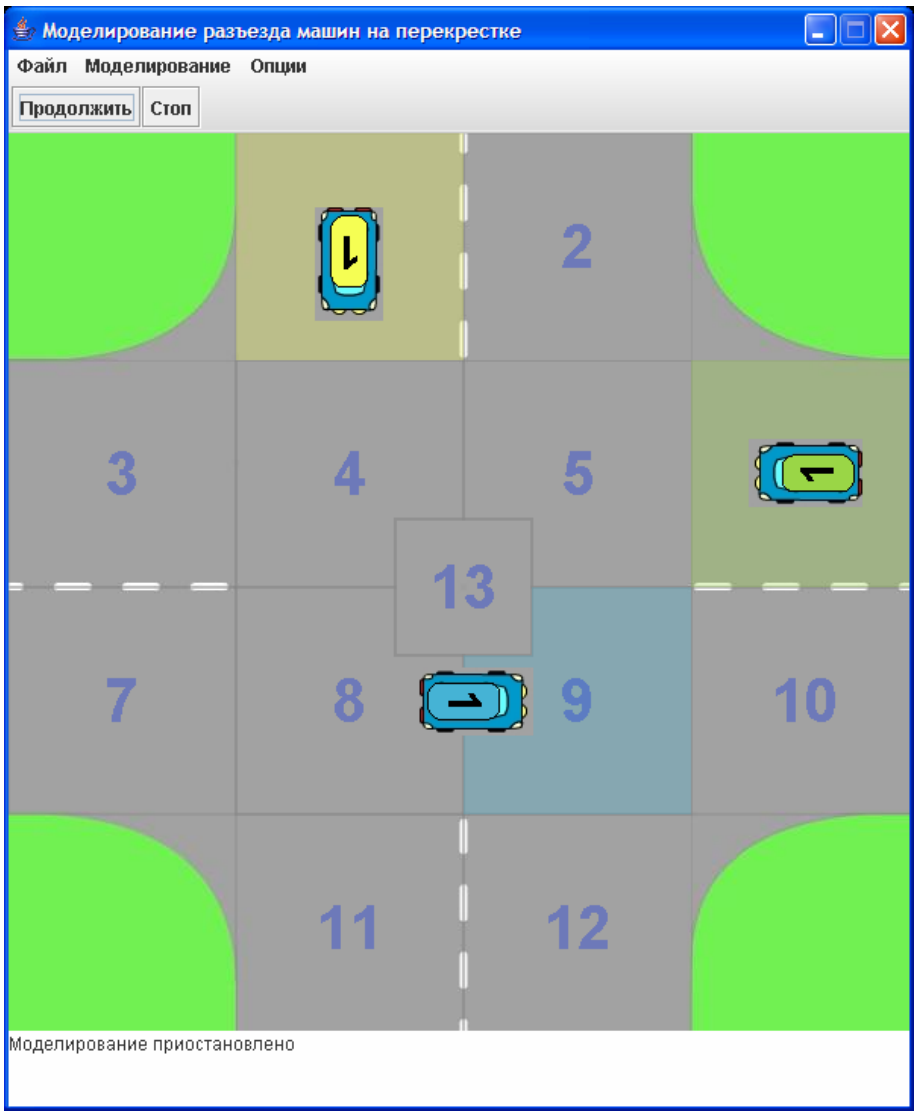

Рис. 14. Пример трансляции сообщений

Cross

Объект `Cross` выполняет три функции:

1. В начале моделирования всем автоматам рассылает сообщения *E2* и *E4*.
2. По тику таймера перемещает машины, находящиеся в состоянии `MOVING`.
3. Транслирует сообщения, сгенерированные объектами `CarCO`, всем остальным машинам.

Событие	Описание
E2	Включить сигналы поворота
E4	Начать движение

Отметим, что сначала все автоматы получают сообщение *E2*, и лишь после того, как все автоматы обработали это сообщение, рассылается сообщение *E4*. Это важно, так как при получении сообщения *E4* автомат должен выбрать траекторию движения, при этом он может руководствоваться только «внешней» информацией о машинах: их расположении и указателях поворота, и ничего не знает об истинных направлениях движения других машин. В частности, автомат не может различить, собирается ли другая машина повернуть налево или развернуться.

Объекты управления

CarCO

Автомат может получить следующую информацию о дорожной обстановке:

- зона, в которой находится какая-либо машина;
- состояние другой машины;
- направление движения данной машины.

Состояния машин кодируются следующим образом:

0 – отсутствие машины;
1 – указатели поворота выключены;
2 – включен левый указатель поворота;
3 – включен правый указатель поворота;
4 – включены и левый и правый указатель поворота (аварийные огни).

Направления движения машин кодируются следующим образом:

0 – назад;
1 – влево;
2 – вперед;
3 – направо.

Входное воздействие	Описание
X1	Указатели поворота машины слева
X2	Указатели поворота машины спереди
X3	Указатели поворота машины справа
X4	Зона, в которой находится данная машина
X5	Направление движения данной машины
X6	Зона, в которой находится машина слева
X10	Зона, в которой находится встречная машина
X11	Зона, в которой находится машина справа

Выходными воздействиями автомата являются:

- управление состоянием указателей поворота;
- управление движением (ехать/стоять);
- подача сигнала другим водителям в случае неопределенности очередности проезда;
- выбор траектории поворота налево или разворота (малая/средняя/большая траектории);
- генерация события по умолчанию.

Последнее действие необходимо, если автомат должен пройти несколько состояний при обработке одного события. Оно является исключительно вспомогательным и служит для упрощения устройства автомата.

Выходное воздействие	Описание
Z1	Стоять
Z2	Включить левый сигнал поворота
Z5	Включить правый сигнал поворота
Z6	Подать сигнал другим машинам
Z7	Ехать
Z9	Выбрать малую траекторию
Z10	Выбрать среднюю траекторию
Z11	Выбрать большую траекторию
Z12	Сгенерировать событие по умолчанию

Автомат Car1

Автомат `Car1` представляет собой правильную реализацию правил проезда нерегулируемых перекрестков равнозначных дорог. На рис. 15 изображена диаграмма его состояний.

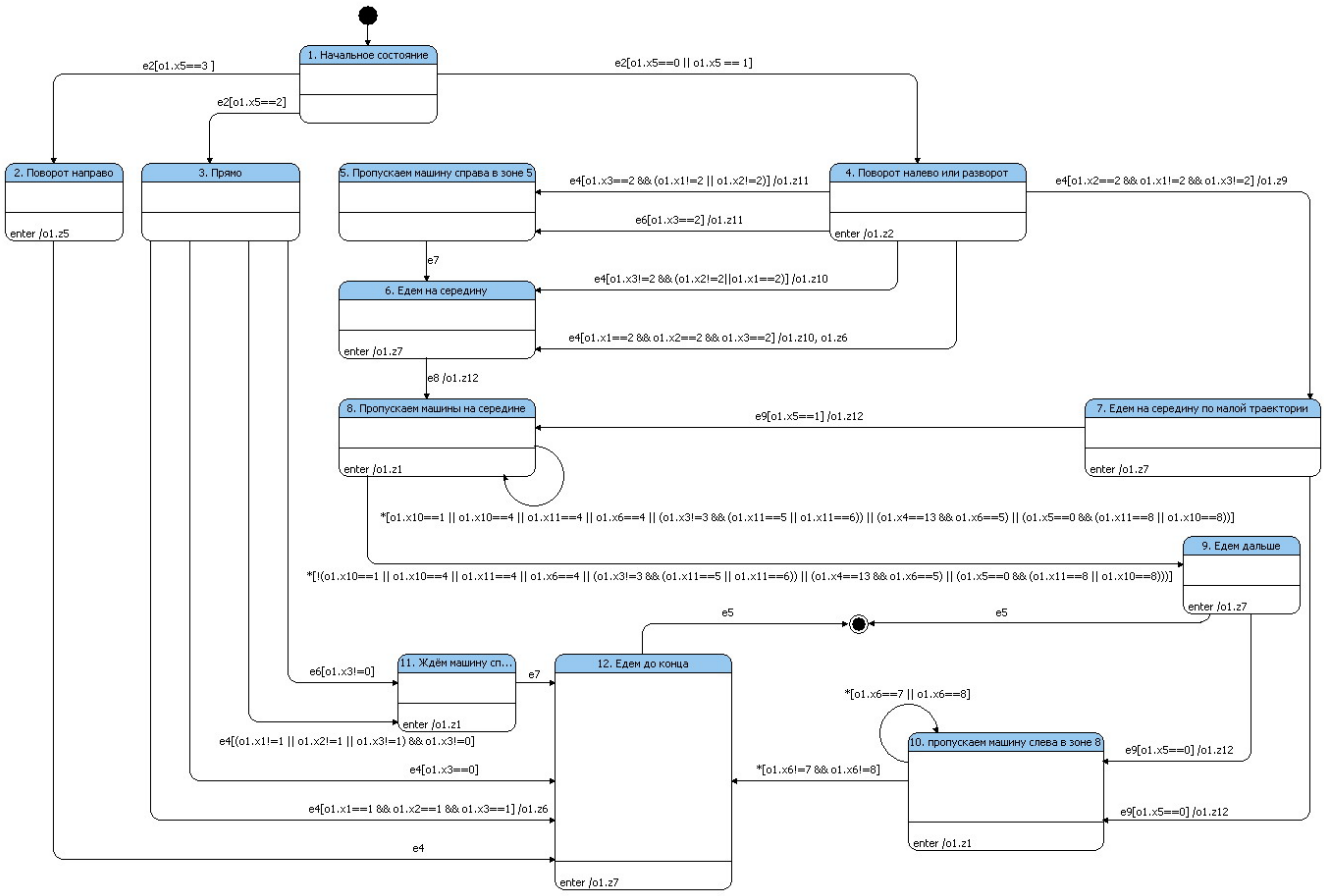

Рис. 15. Автомат `Car1`

Сначала автомат находится в начальном состоянии (состояние 1).

При получении первого сообщения о включении указателей поворота (сообщение *E2* от объекта `Cross`), автомат анализирует выбранное направление движения (`o1.`*X5*) и переходит в соответствующее состояния, включая при этом необходимые указатели поворота.

Рассмотрим три случая:

1. ***Поворот направо*** (состояние 2).
   В этом случае машина никого не пропускает и едет вдоль траектории до конца

2. ***Движение прямо*** (состояние 3).
   Отдельно рассмотрим случай потенциальной взаимной блокировки (ситуации, когда машины не могут начать движение, из-за того, что являются помехами друг другу).
   a. *Все четыре машины намерены двигаться прямо*.
      В этом случае возникает неопределенность и, в соответствии с ПДД, водители должны договориться об очередности проезда. В автомате это реализовано следующим образом: тот автомат, который первым обрабатывает сообщение *E4* «начать движение», в этом случае вызывает действие `o1.`*Z6* «подать сигнал другим машинам» и начинает двигаться (состояние 12). Другие автоматы получают сообщение e6 «другая машина подала сигнал» раньше сообщения *E4* и при его обработке переходят в состояние «ждём машину справа» (состояние 11), а сообщение *E4* игнорируют.
   b. *Иначе.*
      В любой другой ситуации машина должна пропустить машину справа. Поэтому при получении сообщения *E4* «начать движение» автомат проверяет, есть ли машина справа и, если есть, переходит в состояние «ждём машину справа» (состояние 11), а иначе – в состояние «едем до конца» (состояние 12).

   В состоянии «ждём машину справа» (состояние 11) автомат вызывает действие `o1.`*Z1* «Стоять» и ждёт получения сообщения *E7* «машина справа покинула зону 5», после чего переходит в состояние «едем до конца» (состояние 12).

3. ***Поворот налево или разворот*** (состояние 4).
   В дальнейшем, для простоты описания, фразу «поворот налево или разворот» будем заменять на «движение налево».

   В соответствии с ПДД, при движении налево машина должна выехать на середину перекрестка, пропустить машину справа и встречную машину, а если она разворачивается – то и машину слева, и завершить проезд. Конкретная траектория движения машины зависит от направлений движения других машин:
   a. Если встречная машина тоже двигается налево, то имеет место разъезд правыми бортами, то есть машины двигаются одновременно, но так, чтобы не столкнуться. В автомате это соответствует выбору малой траектории (состояние 7).
   b. Если машина справа тоже двигается налево, выехала на середину и остановилась, например, пропуская машину слева, то нужно объехать её сзади. В автомате это соответствует выбору большой траектории (состояние 5).
   c. В остальных случаях выбирается средняя траектория, проходящая через середину перекрестка (состояние 6).

   Так же, как и в случае движения всех машин прямо, возникает неопределенность в случае движения всех четырёх машин налево. Ситуация разрешается аналогичным образом: машина, которая первой получает сообщение e4, подаёт сигнал и начинает двигаться по средней траектории (состояние 6). Остальные машины при получении сообщения e6 переходят в состояние «Пропускаем машину справа в зоне 5» (состояние 5).

Рассмотрим два случая:

a. *Движение по средней или большой траектории* (состояния 5, 6).
Если машина справа двигается налево (а, следовательно, выбрана большая траектория), то необходимо сначала её пропустить, а лишь затем выехать на середину (состояние 5). В случае выбора средней траектории автомат сразу переходит в состояние «ехать на середину» (состояние 6), так как его траектория до середины перекрестка не пересекается с траекторией машины справа.

Считается, что машина доехала до середины перекрестка, когда её центр пересек продольную разделительную линию. В этот момент автомат получает сообщение *E8* «Данная машина пересекла продольную разделительную линию» и переходит в состояние 8 «Пропускаем машины в зоне 4». В этом состоянии автомат пребывает до тех пор, пока не выполнится сложное условие, которое будет рассмотрено далее. Это условие проверяется при получении любого сообщения. Для того, чтобы проверить его при попадании в состояние 8, при переходе из состояния 7 генерируется событие по умолчанию, которое инициирует проверку условия сразу после попадания в состояние 8.

Теперь разберем это условие. Машина должна стоять на месте на середине перекрестка, если верно одно из следующих утверждений:

- Встречная машина находится в зоне 1, то есть в своей начальной зоне.
  Куда бы ни ехала встречная машина, нужно её пропустить
- Какая-либо машина находится в зоне 4.
  Через зону 4 проходит траектория данной машины, поэтому в любом случае нужно дождаться её освобождения
- Машина справа двигается прямо или налево и находится в зоне 5 или 6.
  Траектория машины справа пересекает траекторию данной машины, и она ещё не доехала до пересечения. Надо её пропустить.
- Данная машина находится в зоне 13, и машина слева находится в зоне 5.
  Машина слева или разворачивается, или поворачивает налево. Точнее определить нельзя, так как известно только, что у неё включен левый указатель поворота. Но если она разворачивается, то она будет для данной машины помехой справа, поэтому нужно её на всякий случай пропустить.
- Данная машина разворачивается, а зона 8 занята другой машиной.
  Машина должна дождаться освобождения зоны 8, так как через неё проходит траектория данной машины.

Как только все условия становятся ложными, автомат переходит в состояние 9 «едем дальше». Затем, если данная машина разворачивается (`o1`.*X5* == 0), то при пересечении поперечной разделительной линии она снова останавливается (состояние 10) и пропускает машину слева. После этого завершает проезд перекрестка (состояние 12).

b. *Движение по малой траектории* (состояние 7)
Автомат выбирает малую траекторию, если встречная машина тоже двигается налево, а ни машина слева, ни машина справа не двигаются налево.
В этом случае машина двигается вдоль малой траектории до пересечения поперечной разделительной линии (состояние 7). Затем, если она поворачивает налево, то автомат переходит в состояние 8 «Пропускаем машины на середине», а если разворачивается – в состояние 10 «Пропускаем машину слева в зоне 8». Оба этих состояния уже были рассмотрены ранее.

Автомат Car2

На рис. 16 изображена диаграмма состояний автомата `Car2`.

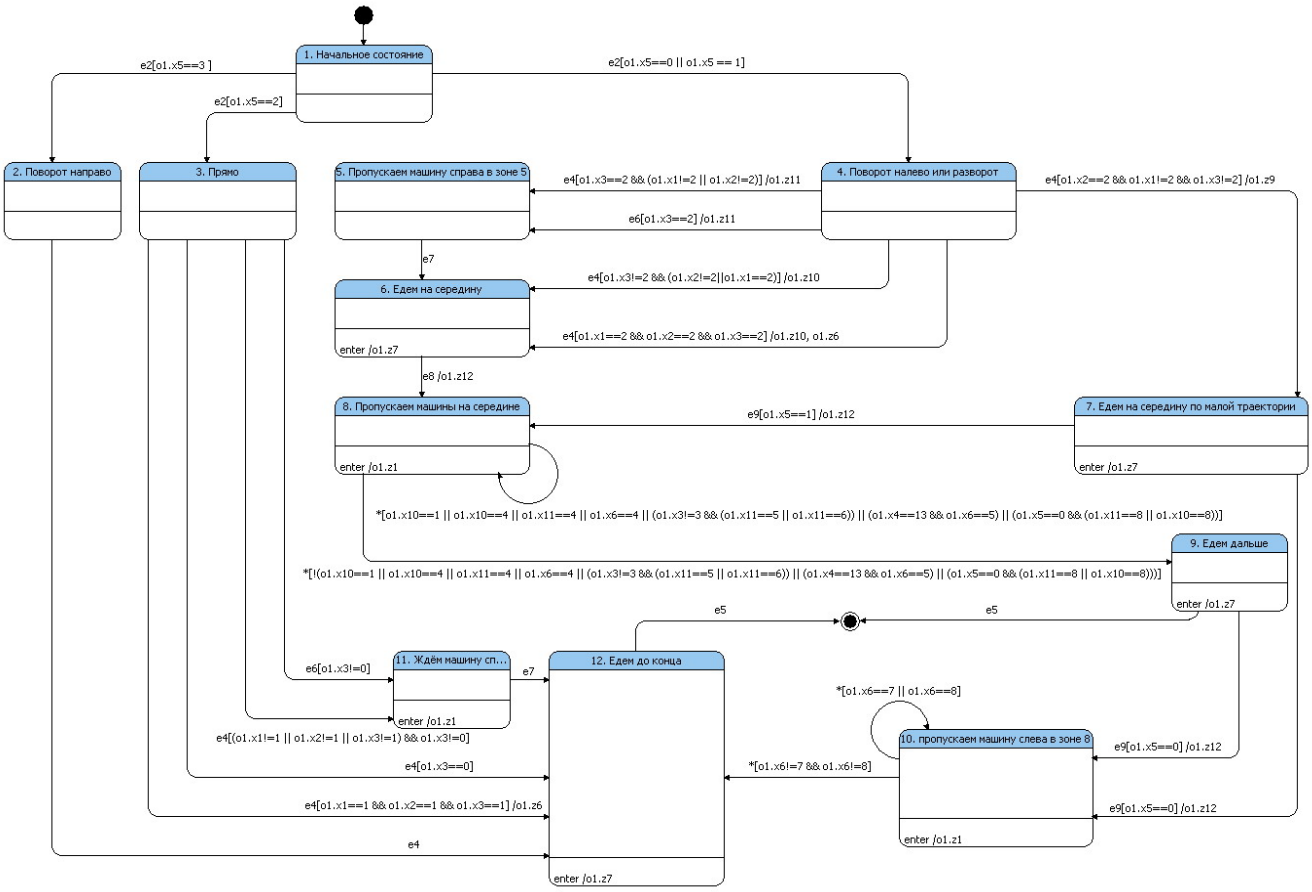

Рис. 16. Автомат `Car2`

Автомат `Car2` является копией автомата `Car1` за исключением того, что он соответствует водителю, который не включает указатели поворота, когда это требуется. Таким образом, автомат `Car2` моделирует действия водителя, который соблюдает правила дорожного движения, но не знает, что у машины не работают указатели поворота, или забывает их включить. Это может привести к аварии, так как остальные участники дорожного движения будут принимать решение о траектории и очередности проезда, полагая, что данная машина движется прямо.

Автомат Car3

Автомат Car3 (рис. 17), напротив, включает необходимые указатели поворота, но совершенно не соблюдает правила дорожного движения, а сразу же начинает движение и никого не пропускает на своём пути. Таким образом, этот автомат имитирует поведение «наглого водителя».

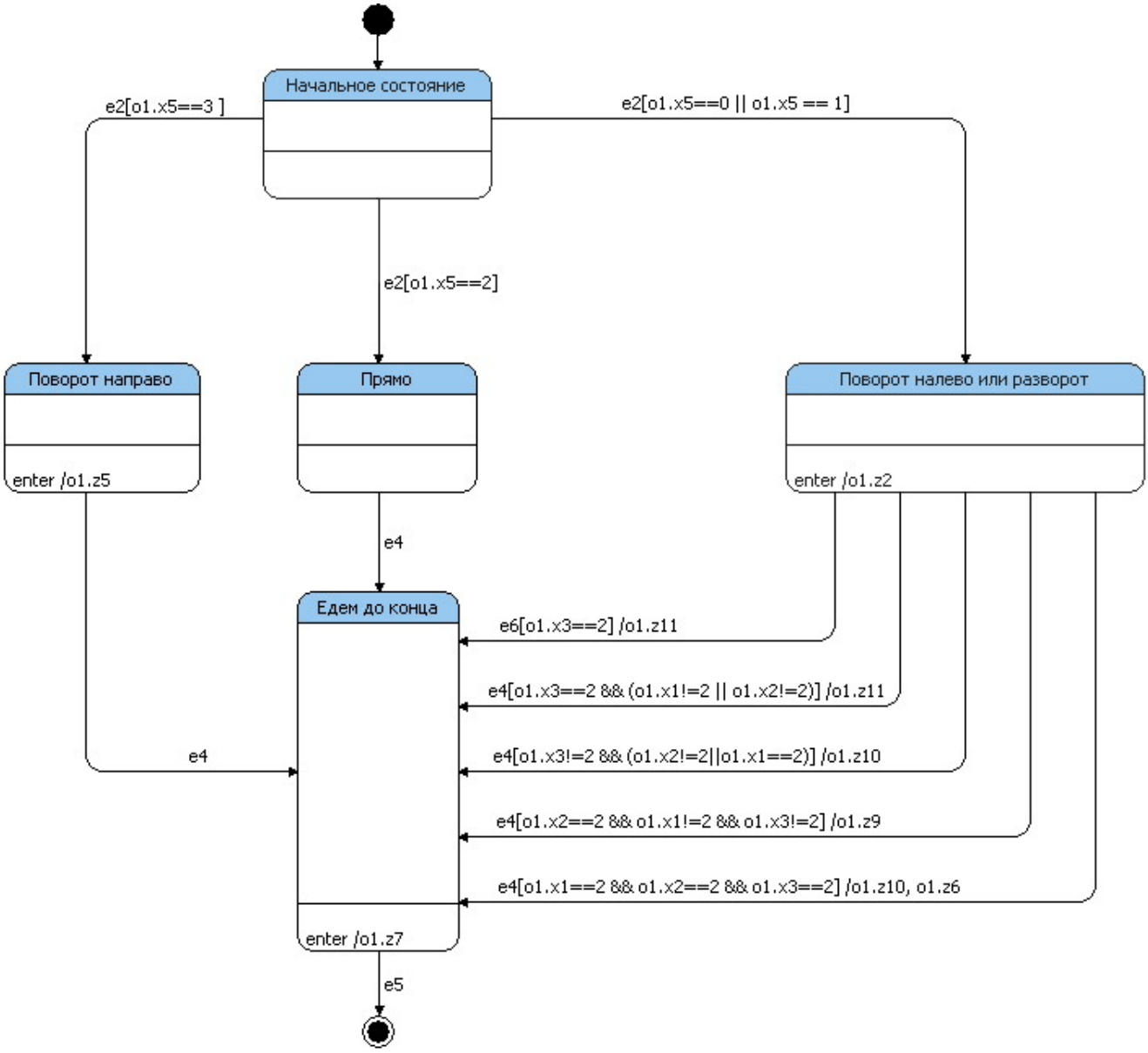

Рис. 17. Автомат `Car3`

Заключение

Возможные пути улучшения проекта состоят в рассмотрении более широких классов: регулируемых перекрёстков и/или перекрёстков неравнозначных дорог; и добавлении других транспортных средств таких, как трамваи и автомобили со спец. сигналами. Однако это приведёт к существенному усложнению автоматов.

Литература

1. *Шалыто А. А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
2. *Паращенко Д. А., Царев Ф. Н., Шалыто А. А.* Технология моделирования одного класса мультиагентных систем на основе автоматного программирования на примере игры «Соревнование летающих тарелок». Проектная документация. СПбГУ ИТМО. 2006. http://is.ifmo.ru/unimod-projects/plates/
3. *Правила дорожного движения*, утвержденные постановлением Правительства РФ от 23.10.93 № 1090 (в ред. от 28.02.2006).