ГЛАВА 11
Режим расписания

11.1 Введение

Концепция расписания — это не попытка замены сценариев MSTS, а альтернативный способ предопределять задачи для всех поездов игровой сессии.
В сценарии, поезд игрока определён явно; все поезда, управляемые компьютером описаны в файле трафика, а статические поезда задаются индивидуально.
В файле расписания же, все поезда заданы одинаковым образом: в структуре данных нет различий между активными поездами – любой из них может быть выбран в качестве поезда игрока, а все остальные будут двигаться под управлением компьютера. Статические поезда, в принципе, задаются аналогичным образом, но недоступны для выбора.
Перед запуском сессии, желаемый поезд выбирается игроком из списка. В итоге, количество различных "сценариев", которые можно воспроизвести с использованием одного и того же файла расписания, равно количеству поездов, заданных в нём, за вычетом статических.
Описания случаев, когда принципы ORTS или MSTS для расписаний и сценариев существенно отличается друг от друга, выделены **жирным шрифтом**.
Концепция расписаний все еще находится в стадии разработки. По мере продолжения работы все элементы по-прежнему могут быть изменены!

11.2 Общие положения

11.2.1 Формат данных

Данные расписания содержатся в электронной таблице и сохраняются в виде файла `*.csv` (файл с символами-разделителями) в формате Unicode. В качестве разделительного символа необходимо использовать: "," (запятая),";" (точка с запятой), либо *символ табуляции*.
Не выбирайте пробел в качестве разделительного символа.
Поскольку ";","," или *tab* являются возможными разделительными символами, ***эти символы не должны использоваться в пределах фактических данных***. Заключение текста в кавычки (одинарные или двойные) не имеет никакого эффекта. Кроме того, ***символ*** "#" ***не следует использовать в названиях поездов***: это префикс для зарезервированных слов в расписании.

11.2.2 Файловая структура

У сохраненных файлов, расширение `*.csv` должно быть заменено на `*.timetable-or`.
Файлы расписания помещаются в подкаталог с именем *OpenRails*, созданный в каталоге сценариев полигона (*Activities*)

11.2.3 Группировка расписаний

Несколько расписаний можно объединить с помощью *файлов-списков групп расписаний*. Тогда, в одной игровой сессии, все они будут загружаться и действовать совместно, включая возможность перекрёстных ссылок между расписаниями группы. Файл-список также должен находиться в подкаталоге *OpenRails* каталога сценариев полигона. Это обычный текстовый файл с расширением `*.timetablelist-or` , каждая строка которого содержит полное имя файла расписания, подлежащего включению в группу.
Если первая строка, начинается с символа "#" - следующий за ним текст будет использоваться в качестве отображаемого имени группы расписаний в меню Open Rails.
Вот пример файла группы расписания:

```
#Все поезда Северо-Восточного Коридора - Пятница Август 2018 год
Amtrak - Пт Авг 2018.timetablelist-or
МАРК Линия Кэмден - Пт Авг 2018.timetable-or
МАРК Линия Пенн - Пт Авг 2018.timetable-or
СЕПТА Уилмингтон-Ньюарк - Пт Авг 2018.timetable-or
```

11.2.4 Файлы пула

Пулы (резерв) можно использовать для хранения поездов, не находящихся в рейсе, в порядке «живой очереди», без необходимости вручную программировать маршруты к путям отстоя и от них. Файлы пула находятся в том же подкаталоге *OpenRails*, что и другие файлы расписания. Они имеют расширение `*.pool-or` или `*.turntable-or`.

11.2.5 Файлы погоды

Файлы погоды - функционал исключительно для режима расписания - программируют изменения облачного покрова, осадков и дистанции видимости в течение дня расписания. Они помещаются в специальный подкаталог *WeatherFiles* корневой папки полигона, и имеют расширение `*.weather-or`. *Игрок активирует файл погоды, выбрав его в разделе «Режим расписания» главного меню ORTS; тогда статические погодные условия из меню переопределяются.*

11.2.6 Выбор расписаний и поездов

Для запуска игры в режиме расписания, в главном меню ORTS выбирается режим «*Расписание*». Затем, в выпадающем списке выбора расписания, необходимо выбрать нужный файл расписания или файл группы расписаний. В последнем случае, в списке «*График*» выбирается, также, и одно из расписаний, входящих в группу.

После выбора требуемого расписания, предоставляется список всех поездов, содержащихся в нём, и можно выбрать желаемый поезд.

Также можно выбрать сезон и погоду (статическую или определяемую файлом); **они не предопределены в файлах расписания**.

11.3 Основы концепции расписания

11.3.1 Общие положения

Расписание - это список поездов, где для каждого из них задано требуемое время движения. Задание может содержать только время старта, или может включать также промежуточное время.

На данный момент, промежуточные тайминги (временные отсечки) применимы только к местоположениям "платформы", созданным с помощью редактора маршрутов MSTS.

Каждый столбец в электронной таблице содержит данные для отдельно взятого поезда, а каждая строка представляет собой местоположение (т.е. *раздельный пункт*). Ячейка на пересечении столбца поезда и строки раздельного пункта содержит, как правило, данные о времени (*тайминг*) для данного поезда в данном местоположении (***далее, Ячейка=Поле***).

Специальные строки и столбцы могут быть добавлены для записи общей информации или *команд управления* поездом. Общие правила таковы:

Первая строка для каждого столбца *содержит имя поезда*.

Первый столбец для каждой строки *содержит имя раздельного пункта*.

Ячейка на пересечении первой строки и первого столбца должна быть пустой.

11.3.2 Типы столбцов

Столбец определяется содержимым его первой строки (*верхнее поле*).

По умолчанию, первая строка содержит имя поезда.

Специальные столбцы могут быть определены с помощью следующего синтаксиса:

- `#comment` : столбец содержит комментарий и игнорируется при загрузке расписания.
- `<пусто>` : столбец является расширением предыдущего столбца.

11.3.3 Типы строк

Строка определяется содержимым её первого столбца (*левое поле*).

По умолчанию, первый столбец содержит имя раздельного пункта (название станции).

Специальные строки могут быть определены с помощью следующего синтаксиса:

- `#comment` : строка содержит только комментарий и игнорируется при чтении расписания;
- `<пусто>` : данная строка-это расширение вышележащей строки;

- `#path` : определяет маршрут поезда;
- `#consist` : определяет состав поезда;
- `#start` : определяет время запуска поезда;
- `#note` : определяет общие примечания и исходные команды управления для поезда;
- `#dispose` : определяет, поведение поезда после завершения рейса;
- `#speed`, `#speedmph` или `#speedkph` : определяет скоростной режим поезда в метрах в секунду, милях в час или километрах в час соответственно; ***в одном файле расписания можно использовать только один вид строки скорости*** .
- `#restartdelay` : определяет рандомизированые задержки для поезда;
- `#briefing` : строка содержит текст пояснения для поезда, отображаемый в меню, и игнорируется при загрузке расписания.

11.3.4 Содержимое ячеек

Каждая ячейка на пересечении столбца поезда и строки местоположения, может содержать сведения о времени для этого поезда в этом местоположении.
Команды синхронизации могут быть заданы как для мест, где поезд имеет остановку, так и для мест, где не вводится время - когда поезд проходит раздельный пункт без остановки.

11.4 Подробная информация о Расписании

11.4.1 Имя расписания

Хотя строки и столбцы с заголовком `#comment`, как правило, игнорируются, ***в ячейке на пересечении первой строки*** `#comment` ***и первого столбца*** `#comment` ***необходимо указать имя расписания***. Оно отображается как название расписания в меню Open Rails и используется в синтаксисе ссылок на поезда из других расписаний *(подробнее-в п. 11.4.7.1)*.

11.4.2 Имя поезда

Имя поезда, заданное в верхней ячейке столбца, ***должно быть уникальным*** для каждого поезда в пределах одного файла расписания. Это имя также используется в синтаксисе команд, содержащих ссылки на данный поезд; *(подробнее в п. 1.4.7.1)*.
Нажатие в игре клавиши F7 отображает выноски в виде: «**`имя_поезда:имя_расписания`**»
Последовательность поездов не имеет значения.

11.4.3 Имя местоположения

В настоящее время, в качестве возможных местоположений принимаются *маркеры платформ*, созданные в редакторе маршрутов MSTS.
Каждое местоположение характеризуется *названием станции*, содержащимся в описании соответствующей платформы.
Выбор местоположений ограничен лишь "платформами" ввиду того, что не все боковые пути полигона имеют в своём описании привязку к конкретной станции и уникальные имена, а текущая логика не располагает способом их однозначной идентификации.
Имя местоположения, задаваемое в файле расписания, должно точно совпадать с именем в базе данных путей полигона (файл `*.tdb`), иначе местоположение не сможет быть найдено, а значит - не будет обработано.
Кроме того, имя каждого местоположения ***должно быть уникальным***, во избежание неоднозначного соответствия задания для поезда его местоположению.
Впрочем, одна и та же станция может быть задана несколько раз. Последовательность строк местоположений не важна, так как порядок проследования станций поездом определяется в маршруте этого поезда. При этом маршрут может быть составлен так, чтобы поезд проследовал некоторые пункты напроход, или обходил определенные станции.

11.4.4 Детали местоположения

Каждая ячейка на пересечении столбца поезда и строки местоположения может содержать сведения о времени стоянки этого поезда в этом месте. Время задаётся в 24-часовом формате как «`чч:мм`». Если задано одно время, оно принимается за время отправления (за исключением

конечного пункта). *Когда необходимо задать время прибытия и отправления, они должны быть разделены символом "-".*

Дополнительно могут быть заданы команды синхронизации. *Эти команды могут задаваться даже для мест, где поезд не останавливается и, следовательно, отсутствуют детали времени, но поезд должен пройти через это место, чтобы команды возымели эффект.*

Хотя строки с именем одного и того же местоположения могут присутствовать в расписании более одного раза, невозможно определить время стоянки поезда в этом местоположении несколько раз. *Если моделируемый поезд, следуя по маршруту, должен несколько раз совершить остановку в одном и том же месте, в расписании такой маршрут надо разделить на отдельные части, каждая из которых будет соответствовать отдельному поезду.*

11.4.5 Специальные столбцы

- `#comment` колонка комментариев.

Столбец с заголовком `#comment` в первой строке является столбцом комментариев и игнорируется при загрузке расписания, за исключением ячейки на пересечении первого столбца комментариев и первой строки комментариев.

- `<Пустой>` столбец.

Столбец с незаполненной (пустой) ячейкой в первой строке считается продолжением предыдущего столбца. *Его можно использовать для записи команд управления, которые применяются к данным из предыдущего столбца. Это может быть полезно, когда тайминги вычисляются автоматически с помощью формул в электронной таблице, поскольку вставка команд в саму ячейку времени исключит использование таких формул.*

11.4.6 Специальные строки

- `#comment` строка комментариев.

Строка с заголовком `#comment` в первом столбце является строкой комментариев и игнорируется при загрузке расписания, за исключением ячейки на пересечении первого столбца комментариев и первой строки комментариев.

- `<Пустая>` строка.

Строка с пустой ячейкой в первом столбце считается продолжением предыдущей строки.

- `#path` строка маршрутов движения. ***Эта строка обязательна к заполнению.***

Строка с заголовком `#path` задаёт маршрут поезда. Здесь указывается имя файла `*.pat` (без расширения), созданного в Редакторе сценариев MSTS, или с помощью Trackviewer и расположенного в каталоге *Paths* полигона.

В режиме расписания используются такие же `*.pat`-файлы, что и для сценариев.

Однако маршруты, используемые в расписаниях, не должны содержать точек ожидания, поскольку обработка точек ожидания не поддерживается логикой концепции расписания. Детали ожидания в расписаниях задаются с помощью специальных команд управления.

Заголовок `#path` может быть дополнен атрибутом `/binary`.

Большие расписания могут содержать много маршрутов движения поездов, и загрузка их всех может занимать значительное время (порядка нескольких минут). Чтобы сократить время загрузки, маршруты можно хранить в обработанном двоичном формате, аналогичном тому, что используется в файлах-сохранениях игр. Обратите внимание, что информация в двоичном формате не может быть напрямую доступна пользователю ни для чтения, ни для записи.

Когда установлен атрибут `/binary`, программа проверит, существует ли двоичный вариант маршрута. Если это так, она прочитает этот вариант. Если нет, она прочитает "нормальный" файл маршрута, а затем сохранит его в двоичном виде для дальнейшего использования. Двоичные файлы маршрутов хранятся в подкаталоге с именем OpenRails, который ***должен быть предварительно создан Вами*** *в каталоге Paths полигона.*

Важно:

– Если маршрут был отредактирован, его двоичная версия должна быть удалена вручную, в противном случае программа продолжит использовать эту устаревшую версию.

– Если полигон отредактирован таким образом, что файл `*.tdb` мог быть изменен, все двоичные маршруты

должны быть удалены.

• `#consist` строка составов. ***Эта строка обязательна к заполнению.***

Строка с заголовком `#consist` задаёт состав поезда. Здесь указывается имя файла `*.con` (без расширения), созданного в Редакторе сценариев MSTS или редакторе состава TSRE5, и расположенного в каталоге *Consists* полигона.

Однако если поезд в текущей сессии управляется компьютером, и он сформирован из другого поезда (см. Ниже), данная информация о составе игнорируется. Взамен, используется состав поезда, из которого он был сформирован. Для поезда игрока, заданный состав всегда используется, даже если поезд сформирован из другого поезда.

Также, возможен более сложный синтаксис задания состава, что позволяет получать поезда в виде сцепов из нескольких составов, в т.ч. в развёрнутом порядке следования.

Общий вид синтаксиса таков:

`состав [$reverse] [+состав [$reverse] [+ ...] ]`

Пример: поезд с локомотивом в голове, использующий один и тот же набор вагонов, следующий в обоих направлениях, может быть задан в виде (требуется два файла состава: `c_loco.con` и `c_wagons.con`*):*

`c_loco + c_wagons` *и для обратного рейса:*

`c_loco $reverse + c_wagons $reverse`

Существенно, что команда `$reverse` *всегда применяется только к указанному слева от него элементарному составу, а не к полной комбинации составов.*

Если тот же поезд иногда имеет несколько дополнительных вагонов (например, в часы пик), он может быть задан таким образом (требуется `c_add.con` *- файл состава из прицепных вагонов):*

`c_loco + c_wagons + c_add` *и для обратного рейса:*

`c_loco $reverse + c_add $reverse + c_wagons $reverse`

Это избавит от необходимости иметь отдельный файл состава для каждого из указанных поездов и, в частности, - от утомительной задачи создания "развёрнутых" составов. При использовании секционных поездов это еще более рационально.

Предположим, есть два секционных поезда, работающих либо как отдельные поезда, либо как сцеп. Обычно, для моделирования всех возможных вариантов требуется шесть отдельных составов, но теперь достаточно только двух: `set_a.con` и `set_b.con`

Синтаксис в строке составов:

`set_a` (*обратный рейс* `set_a $reverse`)

`set_b` (*обратный рейс* `set_b $reverse`)

`set_a + set_b` (*обратный рейс* `set_b $reverse + set_a $reverse`)

Имена файлов состава, содержащие символы `"+"` ***или*** `"$"`***, допустимо использовать в расписаниях, но они должны быть заключены в*** `"< >"` Например:

`<loco+wagon>+<$loco+wagon>$reverse`

• `#start` строка исходных условий рейса. ***Эта строка обязательна к заполнению.***

Строка с заголовком `#start` задаёт время запуска поезда. Время указывается в 24-часовом формате как «`чч:мм`».

Применительно к поездам управляемым компьютером:

– Когда данный поезд формируется из другого поезда, фигурирующего в расписании, указанное время используется только для определения момента, когда новый поезд становится активным.

Применительно к поезду игрока:

– Указанное время, обычно, становится временем старта игровой сессии.

Когда поезд Б формируется из поезда А, фигурирующего в расписании, если поезд А задерживается и не прибыл до заданного в строке `#start` *поезда Б времени, запуск последнего произойдёт только после прибытия поезда А, из которого он должен быть сформирован. Это относится как к управляемым компьютером поездам, так и к поезду игрока. Поэтому, начало сессии может быть отложено.*

В ячейке `#start` дополнительно может быть задан ряд команд запуска.

Более подробную информацию о запуске и движении поездов около полуночи см. в п. 11.4.7.5

• `#note` строка общих условий рейса. **Опциональна**.

Строка с заголовком `#note` задаёт команды управления, применяемые к рейсу поезда в

целом и не связанные с местоположением. Здесь могут задаваться команды поездам, не останавливающимся и не проходящим через какие-либо из имеющихся местоположений.

- *#speed* строка скоростных режимов. **Опциональна**.

В одном файле расписания может присутствовать только один вариант строки:

#speed (м/с), *#speedkph* или *#speedmph*

Строка с заголовком *#speed* задаёт предельную скорость поезда, с целью замедлить его ход до скорости, более низкой, чем это было бы разрешено в ином случае. При этом,

Скорость, указанная здесь, никогда не будет достигнута, если она превышает максимальную скорость, предписываемую знаками ограничения скорости или сигналами, или заданную в файле состава.

В этой строке, также, может задаваться ряд команд скорости (см. п. 11.4.7.7)

- *#restartdelay* строка задержек отправления. **Опциональна**.

Задержки применяются при приведении поезда в движение после остановки; например, на станции или у сигнала. По умолчанию, для всех поездов установлены случайные величины задержек. Их длительность можно дополнительно варьировать с помощью команд задержки, заданных в строке с заголовком *#restartdelay*.

Итоговая величина задержки вычисляется как *фиксированная часть + переменная часть*, где все значения указаны в секундах.

- *#dispose* строка конечных условий рейса. **Опциональна**.

Строка с заголовком *#dispose* определяет, что происходит с поездом, управляемым компьютером, когда он закончил рейс, т.е. достиг конца заданного маршрута. Здесь можно указать, должен ли поезд быть переставлен в другое место, переформирован в другой поезд, а если да, то как и когда. Дополнительные сведения приведены в разделе о командах завершения рейса (см. 11.4.7.8)

Эта строка может отсутствовать в расписании. Поезд, для которого её ячейка не заполнена, удаляется из игры после завершения своего рейса.

*Строка #dispose в настоящее время **не влияет** на окончание рейса для поезда игрока.*

- *#briefing* строка пояснений **Опциональна**.

Строка с заголовком *#briefing* может содержать текст пояснения, характеризующий данный поезд для пользователя. Этот текст появится в главном окне меню Open Rails наряду с описаниями полигона, локомотива и маршрута. В процессе игры, пользователь также может прочитать его на вкладке *"Инструктаж"* окна справки (вызывается нажатием клавиши F1).

Текст в ячейке на пересечении строки пояснений и столбца *#comment* будет характеризовать данное расписание в целом.

*Формат файла расписания *.timetable-or не позволяет тексту в ячейках содержать разрывы строк, но для текста строки пояснений можно использовать HTML-тэги "
" - они будут преобразованы в разрывы строк.*

11.4.7 Команды управления

11.4.7.1 Общие сведения

В случаях, когда требуется сложное взаимодействие поездов и сигнализации, их поведение и действия можно программировать посредством команд управления.

Синтаксис команд

Все команды имеют одинаковый базовый синтаксис. Вот его элементы:

❶ **$** - префикс для слова, являющегося именем команды;

❷ **Синтаксическое имя** - характеризует команду управления;

❸ **Синтаксический параметр** - задаёт значение, связанное с командой.

Не все команды имеют параметры;

❹ **Синтаксический атрибут** (*квалификатор*) - добавляют дополнительную информацию к команде.

Не все команды имеют квалификаторы. Часть атрибутов опциональны, остальные являются обязательными, или обязательными только в сочетании с другими атрибутами;

❺ **Параметр синтаксического атрибута** - для квалификатора может потребоваться значение.

Пример синтаксиса команды:

```
$имя=значение /атрибут=значение
```

Можно задать несколько значений, разделенных знаком `"+"`.

Имейте в виду: любые квалификаторы всегда применяются ко всем значениям.

Ссылка на поезд

Многие команды требуют ссылки на другой поезд. Ссылкой является имя этого поезда, заданное в верхнем поле его столбца. *Далее в тексте: поезд-субъект команды-«А»; целевой-«Б»*

Если целевой поезд находится в отдельном расписании той же группы расписаний, ссылка дается в виде: `"имя_поезда:имя_расписания"`, где имя расписания представляет собой текст на пересечении первых строки `#comment` и столбца `#comment` в том файле расписания.

11.4.7.7 Команды скоростных режимов

Задаются в строке `#speed`. Размерность значений определяется заголовком строки.

- `$max` *(скорость нагона)*

  Синтаксис: `$max=`<значение>

  Задатёт предельную максимальную скорость.
- `$cruise` *(спокойный ход)*

  Синтаксис: `$cruise=`<значение>

  Ограничивает скорость следования по графику.

  Пока текущее опоздание не превышает пороговую величину, заданную как `$maxdelay`, поезд будет следовать с указанной скоростью.
- `$maxdelay` *(предел опоздания)*

  Синтаксис: `$maxdelay=`<мин>

  Задаёт пороговую величину опоздания (в минутах).

  Когда эта величина будет превышена, поезд разгонится до максимальной скорости.
- `$creep` *(скорость подтягивания)*

  Синтаксис: `$creep=`<значение>

  Задаёт скорость приближения к закрытому светофору, или месту остановки.
- `$attach` *(скорость при прицепке)*

  Синтаксис: `$attach=`<значение>

  Задаёт скорость прицепки.
- `$detach` *(скорость при отцепке)*

  Синтаксис: `$detach=`<значение>

  Задаёт скорость отцепки.
- `$movingtable` *(скорость на кругу)*

  Синтаксис: `$movingtable=`<значение>

  Задаёт скорость движения при работе с поворотным кругом.

11.4.7.2 Команды стоянки

Команды стоянки относятся к местоположению и применяются ко всем поездам, чьи остановки прописаны в данной строке раздельного пункта. Они вводятся рядом с названием раздельного пункта: в следующем пустом столбце, или непосредственно в первом - через пробел.

- `$hold` *(удержание)*

  Держит выходной светофор закрытым (т.е. на нём горит запрещающий сигнал) и открывает его за 2 минуты до указанного времени отправления поезда.

  За выходной принимается тот светофор, что находится за пределами маркера платформы (в направлении движения), но все еще остаётся в пределах того же узла маршрута – т.е. между маркером платформы и светофором не должно быть стрелок.

  По умолчанию, запрещающее показание сигнала удерживаться не будет.

- `$forcehold` *(принудительное удержание)*

  Держит закрытым ближайший по направлению движения светофор, даже если он не является выходным для данной платформы, и открывает его за 2 минуты до указанного времени отправления поезда.

  Это полезно в местах со сложным путевым развитием, когда светофоры не находятся непосредственно у торцов платформ, но их преждевременное открытие может привести к задержке других поездов.

- `$forcewait` *(принудительное ожидание разрешающего сигнала)*

  Запрещает поездам отправление, даже если ближайший светофор - с запрещающим показанием - не распознается как выходной для этой платформы.

- `$nowaitsignal` *(не ждать разрешающего сигнала)*

  Разрешает поездам отправление для сближения со следующим светофором.

  По умолчанию, поезд не отправляется от платформы, если ближайший по направлению движения светофор имеет запрещающее показание и распознаётся как выходной. Но есть ситуации, когда это правило не должно применяться. Например, при отправлении с находящегося на перегоне остановочного пункта без путевого развития и выходных светофоров. Ближайшим тогда, является проходной светофор, который может находиться на некотором удалении от платформы. В этой ситуации поезду не нужно ждать разрешающего показания, чтобы отправиться.

  Это также касается грузовых поездов, одиночных локомотивов и путевых машин, которые тоже обычно не ждут открытия светофора, а подтягиваются к нему, чтобы затратить как можно меньше времени на отправление со станции.

- `$stoptime` *(время стоянки)*

  Синтаксис: `$stoptime=n` (n-время в секундах)

  Задаёт время стоянки по умолчанию для этой платформы, переопределяя значение времени стоянки, заданное в базе данных путей полигона.

- `$terminal` *(тупиковая платформа)*

  Изменяет расчет позиции остановки, заставляя поезд останавливаться у торца платформы. *В обычном случае, середина состава остановится в середине платформы*.

  Применять ли команду к данной платформе, и к какому из её торцов надо подтягивать состав, определяется проверкой маршрута поезда:

  - Если платформа находится на начальной секции маршрута, или отсутствуют пересечения от начала до секции, на которой она находится-предполагается, что поезд отправляется с тупиковой платформы. Тогда хвостовой вагон останавливается у ближнего, в направлении движения, торца.

  - Если платформа находится на конечной секции маршрута, или отсутствуют пересечения после секции, на которой она находится, и концом - предполагается, что поезд прибывает к тупиковой платформе. Тогда головной вагон останавливается у дальнего, в направлении движения, торца .

  - Если не выполняется ни одно из этих условий - платформа определяется, как фактически - не тупиковая, и рассчитывается *обычная* позиция остановки.

  Команда `$terminal` может быть задана для станции в целом, или для отдельных поездов. ***Будучи задана для станции, она не может быть отменена для отдельного поезда.*** Однако, следуя описанной выше логике, если она задана для станции, где имеются как тупиковые, так и сквозные платформы, поезда с маршрутами, проходящими вдоль сквозных платформ, будут иметь *обычные* позиции остановки.

- *$closeup (приблизиться к составу)*
  Заставляет поезд А подтягиваться к поезду Б, уже стоящему у платформы. ***Может работать только в случае, когда для поезда А также задана команда синхронизации*** $callon.
- *$closeupsignal (приблизиться к сигналу)*
  Устанавливает уменьшенное расстояние сближения со светофором, чтобы максимально использовать доступную длину платформы.
- *$extendplatformtosignal (поправка в плюс)*
  Помещает позицию остановки за маркер платформы - в непосредственной близости к сигналу. *Иногда маркер платформы расположен ближе к центру, относительно фактического её торца, у которого находится светофор; например, в случае стрелок в пределах платформы. Это приводит к тому, что поезда останавливаются далеко от торца платформы, и их хвостовые вагоны могут блокировать стрелки.*
- *$restrictplatformtosignal (поправка в минус)*
  Помещает позицию остановки до маркера платформы - в непосредственной близости к сигналу. *Иногда маркер платформы расположен за выходным светофором. Когда тот имеет запрещающее показание, поезд остановится перед ним, и если состав длинный, остановка не будет зачтена, как исполненная, поскольку поезд не достиг требуемой для этой платформы позиции остановки.*
- *$keepclear (не занимать)*
  Смещает позицию остановки, чтобы платформа впереди или позади поезда оставалась свободной на длину, указанную в команде, когда надо прицепить или принять другой поезд на ту же платформу.
  Атрибуты:
  /rear=n (n-расстояние в метрах)
  Задаёт место остановки так, чтобы позади поезда осталось свободно минимум n метров платформы. Когда при платформе есть выходной светофор, поезд остановится перед ним, даже если тогда останется свободно менее n метров; кроме случая, когда также установлен атрибут */force*. ***В этой ситуации, маршрут поезда должен продолжаться за выходной светофор.***
  Учтите: поезд никогда не будет двигаться дальше конца своего маршрута.
  /front=n (n-расстояние в метрах)
  Задаёт место остановки так, чтобы впереди поезда осталось свободно минимум n метров платформы. Если при этом, хвост поезда не будет помещаться в пределах платформы, поезд остановится так, чтобы хвостовой вагон находился у торца платформы, даже если тогда останется свободно менее n метров; кроме случая, когда также установлен атрибут */force*
  /force
  Заставляет поезда точно выдерживать заданную длину свободного участка платформы, даже проезжая для этого за выходной светофор (для параметра */rear*), или оставляя хвост не в пределах платформы (для параметра */front*).
- *$endstop (конечная остановка)*
  Завершает рейс поезда на позиции остановки, хотя маршрут продолжается за её пределы (например, при задании $keepclear /rear /force). Позиция остановки принимается за конец маршрута, даже если поезд ещё не достиг его фактической конечной точки.

11.4.7.3 Команды синхронизации

Применяются к конкретному поезду. Могут вводиться в любой временно́й ячейке, т.е. на пересечении столбца этого поезда и строки раздельного пункта, либо в строке общих условий рейса (#note). Команды будут выполняться в этом местоположении и далее (если применимо).
Например, команда $wait может быть задана в поле станции, где у поезда нет остановки. Фактическим местом её исполнения может быть либо сама эта станция, либо разъезд или стрелка где-то за пределами этой станции.

- *$wait (пропуск поезда Б)*
  Синтаксис:
  $wait=<имя_целевого_поезда> /maxdelay=n /notstarted /atstart /owndelay=n
  Заставляет поезд (условно А) ждать прохода целевого поезда (условно Б). Поезд Б может следовать в попутном или встречном направлении.
  Фактически, скрещение/обгон, выполняется перед ближайшим общим для поездов А и Б

участком пути: начиная с места, для которого задана команда $wait; либо - от начала маршрута поезда А, если команда задана в строке #note.
Когда оба поезда отправляются из одной и той же точки, обгон будет выполнен перед следующим за первым, не являющимся общим для поездов А и Б, общим участком пути.
Для поезда А, общий участок будет считаться занятым до тех пор, пока его не проследует поезд Б.
В случаях, когда позиция ожидания находится вблизи от точки старта поезда Б, и есть риск, что ещё до запуска поезда Б, общий участок может быть занят поездом А, последний можно заставить ждать с помощью атрибута `/notstarted`, в противном случае, поезд Б должен уже существовать, чтобы ожидание было в силе.
Команда `$wait` неэффективна для мест, где поезд меняет направление, так как общий участок будет находиться в следующей части маршрута (после разворота); а поиск выполняется только для активной части маршрута. В такой ситуации, поезд должен быть задан как два отдельных: первый в ходу до места разворота, второй - начиная с этого места (ему и задаётся `$wait`).
Параметр команды: имя целевого поезда (Б), - **обязательно**.
Атрибуты команды:
`/maxdelay=n`: n-максимальное время опоздания поезда Б (в минутах), в пределах которого поездом А всё ещё выполняется ожидание.
Величина задержки поезда Б компенсируется временем собственного опоздания поезда А. Например, если `maxdelay=5` (минут), поезд Б опаздывает на 8 минут, но поезд А сам опоздал на 4 минуты, фактическая задержка составит 4 минуты, и поэтому ожидание все еще действенно.
Атрибут ***опционален***; *по умолчанию применяется значение 0*.
`/notstarted`: выполнять ожидание, даже когда целевой поезд (Б) еще не запущен.
`/atstart`: ожидать строго в данном месте, а не на ближайшем не-общем участке пути.
Для случаев, когда маршрут встречного поезда Б заканчивается на том же участке, где начинается маршрут поезда А, и между этим местом и текущим положением поезда Б нет возможности для скрещения.
`/owndelay=n`: n-минимальное время опоздания поезда А (в минутах), за пределами которого начинает выполняться ожидание.
Используется для задержания опаздывающего поезда, чтобы он, таким образом, не вызывал дополнительных задержек других поездов; в частности на однопутных участках.
`/trigger=ЧЧ:ММ` *Экспериментально**: Ожидание в силе только после указанного времени.
`/endtrigger=ЧЧ:ММ` *Экспериментально**: Ожидание в силе только до указанного времени.
****Применять с осторожностью!*** Предпочтительнее пользоваться `/maxdelay` и `/owndelay`

- `$follow` *(следование за поездом Б)*

Синтаксис: `$follow=<имя_целевого_поезда> /maxdelay=n /notstarted /owndelay=n`
Команда аналогична команде `$wait`, но в отличие от неё, применяется к каждому общему для поездов А и Б участку пути, после той части маршрута, которая не была для них общей. Поезд А управляется таким образом, что на каждом участке, где пути поездов снова соединяются после участка, что не был общим, он будет двигаться только в том случае, если целевой поезд Б проследовал это место. То есть, команда работает как `$wait`, повторяемая для каждого такого участка.
Команда применима только для поездов, следующих в попутном направлении.
Когда найдено место обгона, где поезд А должен быть задержан, выполняется специальная проверка, чтобы убедиться, что хвост поезда А не остался на пути поезда Б или, если это так - что поезд Б уже проследовал это место. ***В противном случае, обгон переносится в другое место,*** *чтобы избежать затора, когда для поезда Б не будет возможности обогнать ожидающий его поезд А.*
Параметр команды: имя целевого поезда (Б), - **обязательно**.
Атрибуты команды:
`/maxdelay=n`: n-максимальное время опоздания поезда Б (в минутах), в пределах которого команда все еще применима.
Величина задержки поезда Б компенсируется временем собственного опоздания поезда А.

Например, если `maxdelay=5` (минут), поезд Б опаздывает на 8 минут, но поезд А сам опоздал на 4 минуты, фактическая задержка составит 4 минуты, и, поэтому, ожидание все еще действенно.
Атрибут ***опционален***; *по умолчанию применяется значение 0*.

`/notstarted`: следование будет в силе, даже если целевой поезд (Б) еще не запущен.

`/owndelay=n`: n-минимальное время опоздания поезда А (в минутах), за пределами которого начинает выполняться следование.

Используется для задержания опаздывающего поезда, чтобы он, таким образом, не вызывал дополнительных задержек других поездов, в частности на однопутных участках.

`/trigger=ЧЧ:ММ` *Экспериментально*:* Следование в силе только после указанного времени.

`/endtrigger=ЧЧ:ММ` *Экспериментально*:* Следование в силе только до указанного времени.

****Применять с осторожностью!*** Предпочтительнее пользоваться `/maxdelay` и `/owndelay`

- `$connect` *(согласованная с поездом Б стоянка)*

  Синтаксис: `$connect=<имя_целевого_поезда> /maxdelay=n /hold=h`

  Заставляет поезд А ждать на стоянке прибытия поезда Б, чтобы пассажиры сделали пересадку между поездами.

  Расписания поездов А и Б должны быть согласованы, чтобы пересадка была возможна в данном месте в данное время, а команда `$connect` задаётся в обеспечение пересадки, если прибывающий поезд опаздывает.

  Команда `$connect` ***не блокирует сигнал***. Если маршруты поездов А и Б конфликтуют до того, как поезд Б достигнет станции, необходимо установить дополнительные команды `$wait` или `$hold`, чтобы избежать затора.

  Параметр команды: имя целевого поезда (Б), который требуется ждать, - **обязательно**.

  Атрибуты команды:

  `/maxdelay=n`: n-максимальное время опоздания поезда Б (в минутах).
  Если опоздание поезда Б превышает это значение, поезд А не будет ждать.
  Фактическое значение ***не зависит от времени собственного опоздания поезда А***.
  Этот атрибут и его значение ***являются обязательными***.

  `/hold=n`: n-время (в минутах), на которое продлевается стоянка поезда А, после прибытия поезда Б, чтобы позволить пассажирам осуществить пересадку.
  Этот атрибут и его значение ***являются обязательными***.

- `$waitany` *(не блокировать маршрут)*

  Синтаксис: `$waitany=<имя_маршрута> /both /opposite`

  Заставляет ожидать прохода любого поезда, находящегося на участке заданного маршрута.
  Если установлен атрибут `/both`, ожидание действует в отношении всех поездов, независимо от их направления; в противном случае, осуществляется ожидание только поездов, следующих в попутном целевому маршруту направлении.
  С установленным атрибутом `/opposite`, команда принимает в расчёт только поезда встречного, относительно целевого маршрута, направления.

  Задаваемый маршрут должен иметь общий участок с маршрутом поезда-субъекта, в противном случае - позиция ожидания не сможет быть найдена.

  Команда может задаваться для управления ожиданием у поездов, выходящим за рамки обычных правил сигнализации или взаимоблокировки. Так, она может использоваться для выполнения проверки свободности перегона на более дальней дистанции, чем это делается сигнализацией, для поезда, отправляющегося с бокового пути или парка, чтобы избежать блокировки маршрута какому-либо поезду, находящемуся на перегоне.

  С установленным атрибутом `/both` *её можно использовать в начале тупиковых однопутных участков, чтобы гарантировать, что поезд не выйдет на такой участок с последнего раздельного пункта, когда на участке уже есть другой поезд, поскольку это может привести к необратимым заторам.*

- `$callon` *(приём на занятый путь)*

  Разешает поезду быть поданным на путь, занятый другим поездом.
  Подробнее см. в обсуждении взаимосвязи между сигнализацией и расписанием.

- *$hold*, *$nohold* и *$forcehold*

  Эти команды функционально идентичны одноимённым командам стоянки, но имеют приоритет над ними и применяются только к текущему поезду.

  $nohold (без удержания)

  Отменяет удержание запрещающего сигнала на выходном светофоре именно для данного поезда и имеет смысл, только когда для данного местоположения в целом, задана команда $hold. Для случаев, когда установленная для станции команда $hold не должна применяться к данному поезду.
- *$forcewait (принудительное ожидание разрешающего сигнала)*

  Идентична одноимённой команде стоянки, но применяется только к текущему поезду.
- *$nowaitsignal (не ждать разрешающего сигнала)*

  Идентична одноимённой команде стоянки, но применяется только к текущему поезду.
- *$waitsignal (ждать разрешающего сигнала)*

  Отменяет команду стоянки $nowaitsignal для текущего поезда.
- *$noclaim (не готовить маршрут до открытия сигнала)*

  Запрещает поезду резервировать блок-участки рельсовой цепи, когда он удерживается по сигналу. *То есть, он постоянно будет «последним в очереди» на деятельных пересечениях, всегда отдавая приоритет любому другому поезду.*
- *$activate (активировать)*

  Синтаксис: *$activate=<имя_целевого_поезда_Б>*

  Активирует поезд Б, в зависимости от поля, где вводится для поезда А: или при запуске, или когда последний находится на указанной станции, или по завершении его рейса.

  Неэффективна, если поезд Б подаётся из отстоя! Для поезда Б должна быть задана команда $activated

11.4.7.8 Команды задержек отправления

Задаются в строке *#restartdelay*

Синтаксис (за исключением команды $reverse): *$command [/fix=<f>] [/var=<v>]*

где <f> - фиксированная, а <v> - переменная составляющие временной задержки, в секундах.

- *$new (при запуске)*

  Задаёт задержку отправления после появления в игре.

  По умолчанию, фиксированная часть=0с; переменная часть=10с.
- *$path (на маршруте)*

  Задаёт задержки отправления после остановки у препятствий на пути, таких как закрытые светофоры или стрелки не по маршруту.

  По умолчанию, фиксированная часть=1с; переменная часть=10с.
- *$station (со стоянки)*

  Задаёт задержки отправления после остановок на станциях.

  По умолчанию, фиксированная часть=0с; переменная часть=15с.
- *$follow (при следовании)*

  Задаёт задержки отправления для следования по удалению за другим поездом.

  По умолчанию, фиксированная часть=15с; переменная часть=10с.
- *$attach (при прицепке)*

  Задаёт задержку отправления после прицепки к другому поезду.

  По умолчанию, фиксированная часть=30с; переменная часть=30с.
- *$detach (при отцепке)*

  Задаёт задержку отправления после отцепки части состава.

  По умолчанию, фиксированная часть=5с; переменная часть=20с.
- *$movingtable (на кругу)*

  Задаёт задержку отправления после использования поворотного круга.

  По умолчанию, фиксированная часть=1с; переменная часть=10с.
- *$reverse (при смене головы)*

  Синтаксис: *$reverse /additional=<значение>*

  Задаёт дополнительную задержку при смене направления движения, отражающую время, необходимое машинисту для прохода по (или вдоль) поезда - в другую кабину.

  По умолчанию, составляет 0,5 секунды на метр длины состава.

  Для поездов, которые толкаются локомотивом, маневровых передвижений, или грузовых поездов - рекомендуется установить величину задержки на 0.

11.4.7.4 Команды переформирования

Вводятся в строках раздельных пунктов, или поле `#dispose`

- *`$detach` (отцепка)*

  Синтаксис: *`$detach <параметры отцепки> <параметры формирования>`*

  Детализирует отцепку части поезда.

  Атрибуты отцепки:

  `/power`

  Отцепить локомотив(ы). Система проверит наличие локомотивов в голове и хвосте поезда; если будут найдены оба, переднему будет отдан приоритет. Если на обоих концах нет локомотивов, ничего не отцепится.

  `/leadingpower`

  Отцепить только головной локомотив. Если спереди нет локомотива - ничего не отцепится.

  `/allleadingpower`

  Отцепить все локомотивы, стоящие в голове поезда. Если таких нет - ничего не отцепится.

  `/trailingpower`

  Отцепить только хвостовой локомотив. Если в хвосте нет локомотива - ничего не отцепится.

  `/alltrailingpower`

  Отцепить все локомотивы, стоящие в хвосте поезда. Если таких нет - ничего не отцепится.

  `/nonpower`

  Отцепить все единицы, не являющиеся локомотивами. Программа определит, в каком конце поезда расположены локомотивы, и отцепит все неавтономные единицы с другого конца.

  Если ни в одном из концов нет локомотивов - отцепка будет с хвоста. Если с обоих концов находятся локомотивы - ничего не отцепится.

  `/units=n` (***n может быть <0 или >0, но n=0 не допускается***)

  Отцепить указанное количество единиц.

  Если n>0, единицы будут отцеплены в голове. Если n<0, единицы будут отцеплены в хвосте. Если n превышает фактическую составность поезда - отцепится всё, и в поезде остается одна единица.

  `/consist=<состав>[+<состав>[+...]]`

  Отцепить указанный(е) состав(ы). Для использования имен составов в команде отсоединения см. Примечание об именах состава ниже.

  Отцепляемая часть должна быть крайней с какого-либо конца поезда: переднего, или заднего.

  Если отцеп задан как список составов, последовательность составов в нём должна быть «снаружи-внутрь», т.е. при отцепке спереди, первый состав в списке должен быть передней частью, а при отцепке сзади, первый состав в списке должен быть задней частью.

  Если ни передняя, ни задняя части поезда не соответствуют заданному составу (первому в списке составу) - ничего не отцепится.

  Атрибуты сформированного поезда:

  `/forms=<имя_целевого_поезда>`

  Отцепленная часть формирует указанный поезд.

  `/static`

  Отцепленная часть образует статический состав.

- *`$attach` (прицепка)*

  Синтаксис: *`$attach=<имя_целевого_поезда(Б)>`*

  Прицепить к поезду Б поезд А; ***при этом поезд А перестанет существовать***.

  Нет смысла прописывать что-либо для поезда А, после остановки, где задана эта команда; и содержимое поля `#dispose` также теряет смысл.

  Если поезд Б, к которому должен прицепляться поезд А, отсутствует в местоположении, где должна произойти сцепка, поезд А будет удалён без прицепки. *Поэтому рекомендуется использовать команду `$wait`, чтобы гарантировать, что поезд Б будет находится в нужном месте.*

  Если задан атрибут `/firstin` или `/setback`, то должно быть наоборот: в этом случае для поезда Б должна быть задана команда `$wait`, чтобы гарантировать, что этот поезд действительно первый.

Атрибуты (***действительны только в местах, где поезду задана остановка!***):

/firstin

Поезд А прибывает первым и будет ждать прибытия поезда Б, чтобы выполнить прицепку. Поезд Б может прийти спереди этого поезда через стрелку или с противоположной стороны.

/setback

Поезд А прибывает первым и будет ждать, пока другой поезд подойдет сзади.
Когда прибудет поезд Б, поезд А осадится назад, чтобы выполнить прицепку.
Это не следует использовать, если от поезда Б должен быть отцеплен локомотив, так как поезд А не будет ждать ухода локомотива Б, прежде чем выполнять прицепку.

- *$pickup (подцепка)*

Синтаксис: *$pickup=<имя_целевого_поезда(Б)> /static*

Прицепить к поезду А поезд Б, или статический состав, находящийся в местоположении, где задана команда. ***Поезд Б перестанет существовать***. *Поезд Б прицепляется целиком, никакие изменения не вносятся в составы ни одного из поездов (за исключением случаев комбинации с командой* `$triggers` *в поле* `#dispose`).

Если в данном местоположении отсутствует состав для прицепки, поезд А продолжит рейс исходным порядком.

- *$transfer (передача)*

Синтаксис: *$transfer=<имя_целевого_поезда> /static <параметры передачи>*

Передать единицы поезда А поезду Б, или статическому составу, находящимся в местоположении, где задана команда.

После передачи единиц между поездами - оба поезда продолжат существовать.
В поезде А должен оставаться хотя бы один локомотив, т.е. он не должен подлежать передаче.
Поезд Б не обязательно должен иметь в составе локомотив, или все локомотивы могут быть отцеплены в процессе передачи.

Атрибуты, задающие тип передачи:

/give

Поезд А отдаёт заданные единицы поезду Б.

/take

Поезд А забирает заданные единицы поезда Б.

/keep

Поезд А отдаёт поезду Б все единицы, кроме заданных.

/leave

Поезд А забирает у поезда Б все единицы, кроме заданных.

Атрибуты, задающие единицы для передачи или сохранения в поезде:

/onepower: только один локомотив.

/allpower: все локомотивы.

/nonpower: все единицы, не являющиеся локомотивами.

/units=<n>

Если число отцепляемых от поезда А единиц <n> превышает составность этого поезда, оно уменьшается таким образом, что в поезде останется одна единица.

/consist=<состав>[+состав[+...]]

Имена составов - частей, которые нужно оставить или передать. Должны задаваться в последовательности, и первое (или единственное) имя состава должно соответствовать части в соответствующем конце поезда.

11.4.7.5 Команды запуска

Вводятся в строке исходных условий рейса `#start`; задают детали появления поезда в игре.

- *$create (создание)*

Синтаксис: *$create[=<ЧЧ:ММ>] [/ahead=<имя_целевого_поезда>]*

Создать поезд А в указанное время. Если не указано - появление произойдёт до запуска первого поезда. Поезд будет "статичным" до наступления заданного ему времени запуска. Обычные правила размещения поездов по-прежнему применяются, поэтому поезд не может быть помещен на участок пути, уже занятый другим составом. Однако, пути отстоя часто вмещают несколько составов. Чтобы сделать такое возможным и обеспечить расстановку

поездов в правильном порядке (первый - спереди) - должен использоваться атрибут `/ahead`. Тогда поезд А будет создан впереди поезда Б в направлении своего маршрута.

Когда несколько поездов стоят на одном парковом пути, необходимо позаботиться о корректности заданных ссылок. ***Ссылка всегда должна быть на сзади-стоящий поезд:*** ***два поезда не могут ссылаться на один и тот же поезд через атрибут*** `/ahead`, *так как это может привести к конфликту.*

Также, ***поезд Б***, на который ссылается атрибут `/ahead`, ***должен быть создан до или одновременно с поездом А, использующим эту ссылку.***

Если общая длина всех составов превысит длину пути отстоя, они "выплеснутся" на прилегающие участки.

- `$pool` *(подача из резерва)*

  Синтаксис: `$pool=<имя_пула> [/direction=forward|backward]`

  Подаёт поезд из заданного пула.

  Маршруты поездов, подаваемых из пула, должны начинаться в конце или около конца одного из маршрутов доступа, заданных для этого пула. Если маршрут начинается раньше, чем последний участок маршрута доступа, перекрывающиеся участки должны полностью совпадать.

  Для пулов веерного типа, направление, в котором поезд выходит с поворотного круга, можно задать с помощью атрибута `/direction`. ***По умолчанию, поезд осаживается назад***.

- `$next` *(на следующий день)*

  Переносит запуск поезда на следующие сутки — после 00:00 в конце расписания.

  Используется для запуска поездов после полуночи.

- `$static` *(статический)*

  Синтаксис: `$static [/pool=<имя_пула>] [/ahead=<имя_целевого_поезда>]`

  Создать поезд А как статический. Принимает атрибуты:

  `/pool` Создать поезд в заданном пуле.

  Чтобы в резерве имелись поезда, они должны быть заданы посредством этой команды. Маршрут должен быть одним из маршрутов отстоя заданных для этого пула. Однако поезд может быть создан и на любом другом из путей отстоя данного пула, что определяется логикой пула. Если в пуле создается больше поездов, чем может вместить пул, выдается предупреждение.

  `/ahead` Идентично атрибуту команды `$create`. (см.выше)

- `$activated` *(активируемый)*

  Активировать поезд по команде `$activate` от другого поезда. *Команда* `$activate` *может выдаваться до или после заданного времени запуска этого поезда.* ***Но строго однократно:***

  Поезд может быть активирован только одним из прочих поездов.

11.4.7.6 Команды рейса

Вводятся в строке общих условий рейса `#note`; применяются с момента запуска поезда. Помимо специализированных команд, перечисленных ниже, в этой строке допустимы все команды синхронизации.

Программа использует средние значения ускорения и замедления для всех поездов (разные наборы для грузовых, пассажирских и высокоскоростных поездов). Но такие значения не всегда адекватны, особенно для современных поездов, что может вызывать опоздания при попытках смоделировать реальное расписание.

С помощью команд `$acc` и `$dec` можно корректировать используемые значения. *Отметим, что эти команды задают не фактическое значение, а коэффициент: значение по умолчанию будет умножено на этот коэффициент.* Однако установка более высоких значений ускорения и замедления не означает, что поезда всегда будут ускоряться и замедляться быстрее, в соответствии с установленным значением. ***Большую часть времени, поведение поезда контролируется с помощью физики.*** Но особенно фактор `$dec` имеет важный побочный эффект. *Значение замедления также используется для расчета ожидаемого тормозного пути. Установка более высокого замедления приведет к сокращению потребного тормозного пути, позволяя поезду двигаться с максимально-допустимой скоростью на большей части перегонов.* Это может сильно влиять на время хода. Однако будьте осторожны, чтобы не установить слишком высокое значение: расчетный тормозной путь должен быть достаточным для надлежащего торможения, иначе поезд не сможет вовремя остановиться, что приведет к сбоям и т.д.

Типичное значение команды `$dec` для современного ПС составляет 2 или 3.

- *$acc (коррекция ускорения)*

  Синтаксис: $acc=<n>

  Скорректировать ускорение для поезда. <n> - это множитель для ускорения по умолчанию.
- *$dec (коррекция замедления)*

  Синтаксис: $dec=<n>

  Скорректировать замедление для поезда. <n> - это множитель для замедления по умолчанию.
- *$doo (поезд обслуживается одним машинистом)*

  Отменяет подачу сигнала к отправлению при наступлении времени отправления со станции. В результате, не будет слышно свистка, зуммера и т. п.
- *$forcereversal (не заезжать за входной)*

  Когда производится «смена головы», и на маршруте поезда, ведущем от места разворота, есть светофор, фактическая точка разворота устанавливается, по умолчанию, таким образом, что поезд перед реверсом будет полностью заезжать за этот светофор, чтобы обратное движение контролировалось его сигналом.

  Установка $forcereversal позволит поезду сменить голову, сразу, по свободности участка. Это полезно при маневрах в парках, когда нет необходимости полностью выезжать за входной светофор.

11.4.7.9 Команды уборки

Могут вводиться в поле #dispose, чтобы задавать поведение поезда по завершении его рейса. См. примечания ниже о поведении поезда игрока, когда он формируется из другого поезда командой уборки или когда сам поезд игрока имеет команду уборки.

- *$forms (переформирование)*

  Синтаксис: *$forms=<имя_целевого_поезда>* <атрибуты>

  Задаёт, как из поезда А, по завершении его рейса, должен быть сформирован поезд Б.

  Для поезда Б, состав берётся от завершившего рейс поезда А; содержимое поля #consist ***поезда Б игнорируется. Поезд Б будет "статичным" до времени, заданного в поле*** #start ***для него.*** *Это означает, что он не будет пытаться резервировать блок-участок/воздействовать на сигналы и т.п; и не будет двигаться, даже если не находится на станции.*

  Если поезд А опаздывает, и прибудет позже, чем время запуска поезда Б, запуск последнего также задержится*, но поезд Б сразу же станет активным, как только он будет сформирован.*

  Для поездов с локомотивной тягой можно задать манёвр ***обгона*** состава локомотив(ами), когда поезду Б предстоит отправляться в противоположном направлении. Атрибут /runround требует маршрута, задающего путь, по которому должен пройти локомотив(ы),выполняя этот манёвр. Если поезд имеет более одного ведущего локомотива - все они выполнят обгон. Любые другие моторные единицы внутри поезда не будут перемещены.

  Конкретные правила и условия выполнения обгона составов см. в разделе обсуждение взаимосвязи между сигнализацией и концепцией расписания.

  Если задан манёвр обгона, можно также определить время, когда он должен состояться. Если время не задано, обгон будет осуществляться сразу же по окончании рейса поезда А.

  Параметр команды: *имя_целевого_поезда*, ***обязателен***.

  Атрибуты команды:

  /runround=<имя_маршрута>: - задаёт путь, по которому пойдёт локомотив, выполняя обгон. ***Опционален***; ***если установлен - значение обязательно***.

  Для более точного управления маневром, рекомендуется вместо этого атрибута использовать команды *$detach* и *$attach*.

  */rrtime=*ЧЧ:ММ: задаёт время, когда должен начаться обгон, в 24-часовыом формате. ***Опционален. Нужен лишь для*** /runround, но ***если установлен - значение обязательно***.

  /setstop: задаёт поезду А условия первой остановки поезда Б, чтобы обеспечить остановку первого в правильном месте. Это необходимо, если для поезда А не задано остановок, но формируемый из него поезд Б начинает рейс со станции.

  Чтобы атрибут работал правильно, маршрут исходного поезда должен заканчиваться в зоне платформы отправления формируемого поезда.

Опционален и не принимает никаких значений.

/atstation: конечное положение поезда А рассчитывается так, как если бы он останавливался на станции, где начинает рейс поезд Б, даже если для поезда А не задана остановка на станции.

/closeup: позиция остановки поезда будет близко к концу пути или другому поезду.

/speed=<v>: задаёт максимальную скорость при манёвре обгона, в м/с. ***Может быть использован только с атрибутом*** */runround*.

- *$triggers (превращение)*

  Синтаксис: *$triggers=<имя_целевого_поезда>* <атрибуты>

  Задает, какой новый поезд Б должен быть сформирован из поезда А, когда его рейс завершится. Однако при использовании данной команды поезд Б будет сформирован с использованием заданного ***для него*** состава, а существующий ***состав поезда А будет удален***.

  Параметр команды: *имя_целевого_поезда*, ***обязателен***.

  Атрибуты команды:

  /runround=<имя_маршрута>: - задаёт путь, по которому пойдёт локомотив, выполняя обгон. ***Опционален***; ***если установлен - значение обязательно***.

  */rrtime=*ЧЧ:ММ: - задаёт время, когда должен начаться обход, в 24-часовом формате. ***Опционален. Нужен лишь для*** */runround*, но ***если установлен - значение обязательно***.

  /setstop: задаёт поезду А условия первой остановки поезда Б, чтобы обеспечить остановку первого в правильном месте. Это необходимо, если для поезда А не задано остановок, но формируемый из него поезд Б начинает рейс со станции.

  Чтобы атрибут работал правильно, маршрут исходного поезда должен заканчиваться в зоне платформы отправления формируемого поезда.

  Опционален и не принимает никаких значений.

  /atstation: конечное положение поезда А рассчитывается так, как если бы он останавливался на станции, где начинает рейс поезд Б, даже если для поезда А не задана остановка на станции.

  /closeup: Позиция остановки поезда будет близко к концу пути или другому поезду.

  /speed=<v>: задаёт максимальную скорость для манёвра обгона в м/с. ***Может быть использован только с атрибутом*** */runround*.

- *$static*

  Синтаксис: *$static /closeup*

  Поезд станет "статическим" после окончания рейса.

  Параметры: отсутствуют.

  Атрибут команды: */closeup*: Место остановки будет близко к концу пути или другому поезду.

- *$stable (оборот)*

  Синтаксис:

  $stable /out_path=<маршр_уб.> */out_time=*ЧЧ:ММ */in_path=*<маршр_под.> */in_time=*ЧЧ:ММ */static /runround=*<маршр_обгона> */rrtime=* ЧЧ:ММ */rrpos=*<место_обгона> */forms=*<поездБ> */triggers=*<поездБ> */speed=*<v> */name=*< поездА'>

  Перемещает поезд в другое место до выполнения команд */forms*, */triggers* или */static*. То есть, это их расширенная форма. В случае */forms* или */triggers* поезд может вернуться в то же самое, или в другое место, откуда на самом деле стартует новый поезд.

  Характерно, что в таких случаях поезд должен сделать два движения: *уборку* и *подачу*.

  В случае, когда задан атрибут */forms*, дополнительно может быть выполнен *обгон состава*.

  Если задан атрибут */triggers*, ***замена состава произойдёт в месте отстоя***.

  Любой разворот (развороты) на пути подачи или в конечной позиции - учитываются при построении нового состава таким образом, чтобы поезд был обращен в правильном направлении, когда окажется в конечном пункте маршрута подачи.

  Команда полезна, когда после прибытия поезда А, формирующего затем поезд Б, нужно освободить платформу, чтобы другие поезда могли использовать её до времени старта поезда Б. Также, она применима для уборки поезда на путь отстоя после завершения его последнего рейса и "закрепления" там в качестве статического ПС.

Для каждого передвижения могут быть отдельно заданы сроки; а если время не задано, передвижение будет выполнено сразу по завершении предыдущего передвижения.
Если сроки определены, поезд будет "статичным" с завершения предыдущего передвижения до требуемого времени начала следующего.
Если сформированный поезд имеет действительную остановку на станции, и маршрут подачи из отстоя (`in_path`) завершается в районе платформы первой его остановки, автоматически будет добавлена проверка "`setstop`" (см. атрибут `/setstop` к команде `$forms`).
Параметры: отсутствуют.
Атрибуты:
`/out_path`=<имя_маршрута>: - задаёт маршрут уборки поезда при перестановке в отстой. Начало маршрута должно совпадать с концом маршрута прибывшего поезда.
`/out_time`=ЧЧ:ММ: задаёт время начала уборки, в 24-часовом формате.
`/in_path`=<имя_маршрута>: задаёт маршрут подачи поезда при перестановке из отстоя к месту начала нового рейса. Начало маршрута должно совпадать с концом `out_path`, конец маршрута должен совпадать с началом маршрута нового поезда.
`/in_time`= ЧЧ:ММ: задаёт время начала подачи, в 24-часовом формате.
`/closeup`: Позиция остановки поезда будет близко к концу пути или другому поезду.
`/callon`: Разрешена подача к платформе, даже если путь занят.
Требуется, чтобы для сигнала, прикрывающего платформу, была реализована функция `TrainHasCallOn` ***или*** `TrainHasCallOn_Restricted`.
`/runround`=<имя_маршрута>: - задаёт путь, по которому пойдёт локомотив, выполняя обгон.
(Дополнительные сведения см. в описании команды `$forms`*)*
`/rrtime`=ЧЧ:ММ: - задаёт время начала манёвра, в 24-часовом формате.
`/rrpos`=<место_манёвра>: задаёт стадию "цикла оборота", когда должен выполняться обгон.
Возможные значения:
- *`out`*: обгон произойдет до начала уборки.
- *`stable`*: обгон проходить в месте отстоя.
- *`in`*: обгон будет выполнен по завершении подачи.

`/speed`=<v>: задаёт максимальную скорость при выполнении манёвра обгона в м/с. ***Может быть использован только с атрибутом*** `/runround`.
`/name`=<имя>: Задаёт имя, которое будет присвоено поезду на период оборота (нахождения в отстое и выполнения маневровых передвижений). Это имя отображается в информации F7, в информации диспетчера ИЛС и в окне диспетчера. ***Атрибут можно использовать только с атрибутом*** `/runround`.
`/static`: поезд станет статическим ПС после выполнения уборки.
`/forms` =<имя_поезда>: поезд сформирует целевой поезд после выполнения подачи.
(Дополнительные сведения см.в описании команды `$forms`*)*
`/triggers` =<имя_поезда>: поезд создаст целевой поезд после выполнения подачи. Замена состава на новый произойдёт в месте отстоя.
(Дополнительные сведения см.в команде `$triggers`*)*
Использование атрибутов команды:
В сочетании с `/static`:
- *`/out_path`*: ***обязательно***
- *`/out_time`*: ***опционально***

В сочетании с `/forms`:
- *`/out_path`*: ***обязательно***
- *`/out_time`*: ***опционально***
- *`/in_path`*: ***обязательно***
- *`/in_time`*: ***опционально***
- *`/runround`*: ***опционально***

- */rrtime*: ***опционально***, ***допустимо только вместе с*** `/runround`
- */rrpos*: ***обязательно, если задано*** `/runround`; ***иначе - недопустимо***

В сочетании с `/triggers`:

- */out_path*: ***обязательно***
- */out_time*: ***опционально***
- */in_path*: ***обязательно***
- */in_time*: ***опционально***

- *$pool (отстой)*

  Синтаксис: *$pool=*<имя_пула> [*/direction=forward|backward*]

  Убирает поезд в заданный пул, по завершении рейса.

  Для пулов веерного типа направление, в котором поезд выходит с поворотного круга, можно задать с помощью атрибута `/direction`. ***По умолчанию, поезд развернется.***

$attach

Идентична одноимённой команде формирования.

$detach

Идентична одноимённой команде формирования.

$pickup

Идентична одноимённой команде формирования.

$transfer

Идентична одноимённой команде формирования.

$activate

Идентична одноимённой команде формирования.

11.5 Дополнительные примечания к расписаниям

11.5.1 Статические Поезда

Статический поезд задаётся установкой `$static` в верхнем поле (т.е. как "имя" этого поезда). Состав и маршрут по-прежнему необходимы: маршрут используется для определения места стоянки состава (задний конец состава-в точке начала маршрута). Время запуска не требуется: состав будет создан с самого начала расписания. Его использование в рамках расписания ограниченно: на него невозможно ссылаться ни в одной команде и т.д., так как у него нет имени.

Прицепка к статическому поезду - только посредством команд формирования; подробности см. ниже.

Существуют некоторые различия между режимами расписания и сценария, в том, как генерируются статические поезда:

В режиме сценария поезд является экземпляром класса Train с типом STATIC.

В режиме расписания поезд является экземпляром класса TTTrain (как и все поезда в режиме расписания) с типом AI, движение AI_STATIC. Это различие может привести к различному поведению в отношении эффектов звука, дыма и света.

11.5.2 Обработка команды `#dispose` для поезда игрока

Когда поезд игрока завершает рейс, и для этого поезда задана команда `#dispose`, чтобы сформировать другой поезд (либо *$forms*, *$triggers*, или *$stable*) – будет действительно сформирован следующий поезд, как задано, и этот следующий поезд теперь будет новым поездом игрока. Таким образом, игрок может продолжить сеанс (движение) на этом новом поезде, например, в обратном направлении.

При формировании нового поезда, «сервис» станет ***неактивным***. Это новое состояние, в котором поезд не имеет права двигаться. Информация на мониторе пути F4 не обновляется, когда поезд неактивен. На дисплее следующей станции в мониторе графика F10 будет отображаться подробная информация о том, когда поезд должен стартовать. Поезд станет активным в момент старта, заданный для сформированного поезда. Для получения точной информации, в окне монитора графика отображается название поезда игрока.

11.5.3 Окончание игровой сессии

По завершении выполняемой по расписанию поездки, программа не будет автоматически закрыта - игрок сам выполняет выход из программы.

11.5.4 Расчет отклонений от графика

Приблизительное время отклонения постоянно обновляется. Это приближение вычисляется по графиковому времени прибытия на следующую станцию. Если текущее время отличается от графикового времени прибытия, и эта разница превышает текущую задержку, задержка устанавливается на эту разницу. Время, необходимое для достижения этой станции, не учитывается.
Это приближение приведет к лучшему регулированию там, где используются параметры `/maxdelay` или `/owndelay`.

11.5.5 Отсутствие автоматической сцепки

В программе есть логика, которая для любого остановленного поезда проверяет, достаточно ли он близок к другому поезду для прицепки к нему. Именно эта логика позволяет поезду игрока сцепляться с любым статическим ПС.
Однако эта логика содержит некоторые действия, которые не соответствуют обработке поездов расписания. Поэтому сцепка поездов в режиме расписания невозможна, за исключением маневров, явно указанных с помощью команд, таких как `$attach` и `$detach`.

11.5.6 Использование Составов в маневровых командах

Любая единица подвижного состава в игре должна быть введена в качестве "нового" поезда. Когда генерируется новый поезд, он формируется так, как задано в строке состава этого поезда.
Каждый вагон будет помнить свой "исходный состав" на протяжении своей "жизни" в игровой сессии. Имя этого "исходного состава" может быть использовано в любой команде `$detach` или `$transfer`, даже если соответствующая часть изменила поезда.
Так, например, если помещается грузовой поезд, который состоит из нескольких частей, каждая из которых имеет свое собственное название состава (используя задание множественного состава), каждый вагон в этом поезде всегда будет помнить свой исходный состав. Когда этот поезд разбирается на части, части передаются в другие поезда и т.д., изначальное название состава все еще может использоваться.
При использовании этого средства важно следить за тем, куда и в каком поезде перемещаются различные части. Поскольку список составов должен быть определен в правильной последовательности, также важно отслеживать конфигурацию сформированных поездов. Преимущество этого метода заключается в том, что не нужно вести подсчет количества единиц в каждом поезде и каждой порции.
Информация о составе не может быть использована, если поезд запущен в пуле, когда этот пул может содержать разные составы. В этой ситуации не определено, какой состав будет формировать фактический поезд.

11.5.7 Требования к сигнализации и концепция расписания

Общие

Концепция расписания более требовательна к работе системы сигнализации, чем "классические" сценарии. Причина - в интенсивном трафике: большое число поездов движется в обоих направлениях, включая идущие в попутном направлении впереди поезда игрока. Существует очень мало сценариев с подобным сюжетом: очевидно, нет смысла задавать поезда, которые никогда не будут видны, но также и потому, что MSTS не всегда может должным образом справиться с такой ситуацией.
Любые недостатки в сигнализации, например, слишком ранняя подготовка маршрута - сразу повлияют на ход сессии. Так, открытие сигналов на однопутном участке слишком далеко вперед, означает резервирование маршрута через несколько разъездов сразу, что приведёт к очень долгому ожиданию для поездов встречного направления. Это, в свою очередь, может вызывать остановку движения, поскольку сразу несколько поездов выполняют скрещения на одних и тех же разъездах.
Подобные ситуации могут возникать на крупных, деятельных станциях: если маршрут через них готовится поезду слишком заранее - другие прибывающие и отправляющиеся поезда будут остановлены. Если команды `$forms` или `$triggers` используются для связки встречных поездов, проблема усугубляется, поскольку любые задержки прибывающего поезда будут сказываться на обратном рейсе.

Маневровые сигналы

Системы сигнализации могут допускать заход поезда на занятый другим поездом путь/блок-участок (так называемая "разрешительная работа").
Разница между "маневровым" и "условно-разрешающим" сигналами (аспекты `STOP_AND_PROCEED`)

заключается в том, что последними также допускается проследование, если поезд впереди движется (в попутном направлении) по блок-участку. А "маневровый", как правило, даётся лишь в том случае, когда поезд, уже занимающий блок-участок, неподвижен.
Поезда, управляемые компьютером, всегда будут заезжать за маневровый светофор и приближаться на заранее заданное расстояние к поезду, занимающему блок-участок.
В районах станций это нежелательно: если поезда могут заходить на уже занятые пути, притом, что общая длина их составов намного превышает длину приёмо-отправочного пути, очередной поезд перекроет горловину станции, блокируя маршруты остальных поездов. Вероятно, это приведёт к полной остановке всего движения на станции и вокруг нее.
Во избежание этого, в границах станций необходимо запретить приём на занятый путь, даже если сигнализация допускает такой режим. Когда же расписанием требуются маневровые передвижения поезда (не важно, управляемого игроком, или компьютером), возможно сделать для него исключение, задав команду `$callon`. В случае если поезд должен прицепиться к другому поезду у платформы, функция активируется автоматически.
Поскольку в MSTS поезда трафика не могут должным образом сближаться при приёме на занятый путь, большинство моделей сигнализации не поддерживает маневровые сигналы, полагаясь, вместо этого, на использование "запросов разрешения". Поезда, управляемые компьютером, не могут выдать этот запрос, поэтому в таких системах `$callon` не будет работать.
В подобных случаях команды прицепки также могут не работать в границах станций. *Обгон состава также требует наличия "маневрового сигнала" для завершающего перемещения локомотива обратно к поезду - чтобы прицепиться к нему. Таким образом, обгон состава, при выполнении в границах станций, также может работать только в том случае, если в сигнализации есть маневровый режим.*
Специальные параметры сигнализации разработаны для адаптации сигналов к функциям, описанным выше. Их можно задать для соответствующих сигналов в скриптах файла `sigscr.dat`
Параметр “`TRAINHASCALLON()`” вернет "`true`", если блок-участок между данным и следующим сигналами содержит платформу, на которой поезд имеет остановку, и для поезда установлен флаг "`callon`". Этот параметр также вернет "`true`", если на участке за сигналом нет платформы.
Параметр “`TRAINHASCALLON_RESTRICTED`” возвращает "`true`" в аналогичных условиях, за исключением того, что он всегда возвращает "`false`", если на блок-участке за сигналом нет платформы.
Обе функции должны использоваться в сочетании с `BLOCK_STATE = BLOCK_OCCUPIED`.

Команды ожидания и маршруты скрещений

От места, где заданы команды `$wait` или `$follow`, выполняется поиск ближайшего общего участка для обоих поездов, начиная с места, где их пути не являются общими. На однопутных линиях с разъездами, где для обоих поездов заданы "обходящие пути", основной маршрут поездов пойдёт по одним и тем же путям на разъездах, и не будет найдено никаких «не-общих» участков. Тогда команда ожидания не сможет выбрать место для ожидания поезда, и процедура не будет работать.
При использовании на однопутных линиях точек ожидания, для их правильной работы, маршруты поездов должны проходить по разным путям.
Автор расписания имеет выбор: либо жёстко задать места скрещения посредством команд ожидания, либо позволить программе определять места скрещения, используя обходные пути.

Команды ожидания и условно-разрешительные сигналы

Команды `$wait` и `$follow` действуют через параметр ‘`blockstate`’ (состояние блок-участка) системы сигнализации. Если в том месте, где поезд должен ждать, используются условно-разрешающие сигналы, и эти сигналы позволяют проследование на занятый блок-участок (`blockstate JN_OBSTRUCTED`), команды `$wait` или `$follow` не будут эффективны, так как поезд не будет остановлен.

11.5.8 Переход через полночь

Расписанием можно охватить полные сутки, поэтому там могут быть поезда, работающие до и после полуночи.

Для поезда игрока действуют следующие правила:
• Поезд, с отправлением до полуночи, будет запущен в конце дня, но продолжит рейс, который должен завершиться после полуночи.
• Поезд, по плану формируемый из другого поезда до полуночи, НЕ будет запущен, если формирующий его поезд опоздает, в результате чего, время формирования сдвинется на следующие сутки. В этой ситуации сессия прерывается.
• Поезда будут запущены в начале следующих суток, только если задана команда `$next;` иначе запуск произойдёт в начале текущих суток.
Для поездов, управляемых компьютером, применяются следующие правила:
• Поезда, которые отправляются до полуночи, будут запущены в конце дня, но продолжат рейс, который должен завершиться после полуночи.
• Поезда, по плану формируемые из других поездов до полуночи, будут запущены, даже если формирующий их поезд опоздает, в результате чего, время формирования сдвинется на следующие сутки.
• Поезда будут запущены в начале следующих суток, только если задана команда `$next;` иначе запуск произойдёт в начале текущих суток.

11.5.9 Просмотр других активных поездов в Расписании

Для отображения на камерах наружных видов другого поезда - нажмите `<Alt+F9>`, чтобы вызвать список активных поездов и выбрать мышью нужный поезд; или нажмите `<Alt+9>`, для циклического перенаведения камер на активные поезда, как описано в разделе «Переключение видов».

11.5.10 Известные проблемы

• Когда команды `#dispose` применяются к поезду игрока, и новый поезд отправится в противоположном направлении, реверсор "перескочит" в положение «назад» при формировании такого нового поезда.
• Команды `#dispose /runround`, еще не могут быть обработаны.
Будет необходимо переключиться на ручной режим АЛС для выполнения этого манёвра.
• Если два поезда с маршрутами, направленными противоположно, предполагается разместить на одном и том же пути с использованием команды `$create /ahead`, они могут быть созданы в неправильном порядке.
• Если для `#path` установлен атрибут `/binary`, но подпапка `OpenRails` в каталоге `Paths` полигона ещё не существует, программа не сможет загрузить какие-либо маршруты.

11.6 Резервирование поездов посредством пулов

Пулы используют для отстоя поездов: до и после выполнения всех рейсов, а также в промежутках между ними. Поезда могут быть в резерве с момента начала расписания. Когда нужно, поезд можно подать из резерва, а по окончании рейса - убрать в резерв.
Нет необходимости ни точно задавать места отстоя поезда, ни распределять очерёдность рейсов, чтобы избежать блокировки одних поездов другими, которые потребуются только в более позднее время. При использовании пулов, система позаботится о фактическом месте отстоя и выберет первый доступный поезд, когда потребуется состав.
Пул состоит из одного или нескольких путей отстоя составов. Также должны быть заданы маршруты доступа. (подробнее см. ниже.) *Особый случай - пул веерного типа, где все пути отстоя примыкают к поворотному кругу. Маршруты доступа - также ведут к нему. При подаче/уборке, поезд заходит на круг, и будет повернут в требуемое положение.*
Резервировать можно любой активный поезд расписания. Если игрок выбрал поезд, подаваемый из резерва - первый доступный состав будет назначен в качестве поезда игрока. Когда поезд, управляемый игроком, убирается в резерв, его рейс заканчивается в отстое. Игрок может остаться с поездом до его следующего рейса, но нет никакого способа предсказать, что это будет/когда это произойдёт, так как всё зависит от других действий, связанных с этим пулом.

11.6.1 Ограничения

Пул может содержать только поезда, которые эквивалентны по использованию. Поезда не обязательно должны быть одного типа, но их использование должно быть взаимозаменяемым. Невозможно выбрать конкретный поезд из резерва.

Прицепка, отцепка или передача вагонов невозможны для поездов, находящихся в резерве. Только фиксированные составы (один или несколько локомотивов или СМЕ) могут быть поданы из пула или убраны в него. Если требуется состав из нескольких секций, они должны быть извлечены отдельно и сцеплены после выхода из резерва. Если в пул необходимо отправить состав из нескольких секций, они должны быть расцеплены перед уборкой в пул. Раз переформирования состава невозможны, пулы могут использоваться только под локомотивы и СМЕ, т. е. для единиц, которые могут двигаться самостоятельно. ***Пулы не могут использоваться для несамодвижущихся вагонов или поездов без локомотива.***

"Переполнение" пула может произойти, когда поезд убирается в резерв, но пути отстоя полностью заняты. В этой ситуации поезд остановится в точке доступа к пулу и будет удален.

"Дефицит" пула может возникнуть, когда поезд запрашивается из резерва, но пути отстоя пусты, и там нет ни одной доступной единицы ПС. В этой ситуации, если для данного пула установлен флаг "`force creation`", поезд будет создан, и рейс начнется в точке доступа. Если этот флаг не установлен, поезд отменяется. В случае дефицита пула в файл журнала записывается предупреждение.

11.6.2 Файлы пула

Пулы задаются в файле, по формату аналогичном файлу расписания, т. е. в электронной таблице *.csv, сохраненной в виде текстового файла с кодировкой unicode.

Файлы должны храниться в том же каталоге, что и файлы расписания:
(<route>\Activities\OpenRails).

Синтаксис файла пула значительно отличается от файла расписания: `Параметр;Значение`

Имена всех параметров находится в первом столбце, и в каждой строке может быть задано только одно значение. Самая первая строка игнорируется.

Расширение файла для обычных пулов - `*.pool-or;` для пулов веерного типа - `*.turntable-or`.

Файл может содержать повторяющиеся параметры для определения нескольких пулов, которые могут не быть связаны друг с другом каким-либо образом.

Существуют некоторые ключевые различия между обычными пулами и пулами веерного типа:

• Для пулов без поворотных кругов, каждый путь отстоя должен иметь, по крайней мере, один маршрут доступа; для пулов веерного типа, маршруты доступа не привязаны к путям отстоя.

• Для обычных пулов, пути отстоя задаются в исходящем направлении; для пулов веерного типа, стойловые пути задаются, как ведущие от поворотного круга - т.е. в направлении стойла.

11.6.3 Пулы без поворотных кругов (парки путей отстоя)

Параметры для пулов без поворотных кругов:

`#comment` Только комментарий. *Значение строки игнорируется.*

`#name` Имя пула. *Это имя должно использоваться в командах расписания* `$pool` для инициализации, подачи или уборки поездов с использованием данного пула.

Эта строка является обязательной и должна предшествовать всем другим строкам.

`#storage` Маршрут, задающий путь отстоя.

Необходимо задать, по крайней мере, один путь отстоя.

Маршрут должен быть проложен в исходящем направлении - это направление поезда, когда он покидает пул.

Путь отстоя должен содержать только один блок-участок; он не может проходить через стрелки или крестовины.

`#access` Маршрут, задающий доступ к пути отстоя.

За каждой строкой, задающей путь отстоя должны следовать одна или несколько строк, где

задаётся маршрут доступа.
Маршрут должен быть проложен в исходящем направлении - это направление поезда, когда он покидает пул.
Путь доступа может проходить через стрелки или крестовины, но ***не должен содержать точек реверса***.
`#maxunits` Для каждого пути отстоя можно определить максимальное количество единиц, которые могут храниться на этом пути. Эта **строка является опциональной**.
Это только ограничение максимального количества единиц. Эффективное число может быть меньше, если длина пути отстоя недостаточна для хранения этого количества единиц.
`#settings` Содержит специальные флаги для использования пула. На сегодня, разрешено только одно значение: `force creation`, которое заставляет поезда появляться на точке доступа, если пул пуст.

Требования к маршрутам доступа
Невозможно задавать “сквозные” парки путей отстоя: маршруты доступа к путям отстоя могут быть заданы только с одного конца пути отстоя, и поезда всегда будут входить и выходить из пула в одном и том же месте.
Хотя каждый путь отстоя имеет свой собственный(е) маршрут(ы) доступа, желательно, чтобы все маршруты доступа заканчивались в одной и той же точке, т.е. все пути отстоя были доступны из этого местоположения. Можно задать несколько точек доступа, но тогда всё равно желательно, чтобы все пути отстоя были доступны из всех точек.
Если из точки доступа можно получить доступ только к части путей отстоя, существует риск того, что поезда могут не распределяться должным образом по всему парку отстоя. В худшем случае, если все поезда всегда отправляются в одну точку доступа и всегда извлекаются из другой точки доступа, и эти точки не имеют доступа ко всем путям отстоя, может возникнуть непрерывная серия “переполнения” и “дефицита” пула, поскольку единицы, отправленные в пул, не могут быть извлечены.

11.6.4 Пулы с использованием поворотных кругов (веерные депо)
Параметры для пулов на основе поворотных кругов:
`#comment` Только комментарий. *Значение строки игнорируется.*
`#name` Имя пула. *Это имя должно использоваться в командах расписания* `$pool` для инициализации, подачи или уборки поездов использованием данного пула.
Эта строка является обязательной и должна предшествовать всем другим строкам.
`#worldfile` Имя *.w-файла полигона, в котором описывается данный поворотный круг.
`#uid` UID поворотного круга в *.w-файле. Вместе с `#worldfile` это идентифицирует поворотный круг, обслуживающий данный пул.
Значения `#worldfile` и `#uid` должны совпадать с соответствующими значениями в файле `turntable.dat`, который определяет действующие поворотные круги.
`#storage` Маршрут, задающий стойловый путь. ***Должен быть направлен от поворотного круга*** . *Должен быть задан хотя бы один стойловый путь.*
Точка старта маршрута должна находиться за пределами области поворотного круга. Стойловый путь ***может содержать только один участок***; он не может проходить через стрелки или пересечения.
`#access` Маршрут, задающий доступ к стойловым путям. ***Должен быть направлен от поворотного круга***. *Должен быть задан хотя бы один маршрут доступа.*
Маршрут доступа не связан с конкретным стойлом, но применяется ко всем стойловым путям, доступ к которым осуществляется через поворотный круг.
Точка старта маршрута должна находиться за пределами области поворотного круга. Маршрут доступа может проходить через стрелки или пересечения, но не должен содержать никаких точек разворота.

#maxunits Для каждого стойлового пути можно определить максимальное количество единиц, которые могут храниться в этом стойле. **Это поле опционально**.
Определяется только максимальное количество единиц. Эффективное число может быть меньше, если длина стойлового пути недостаточна для хранения этого количества единиц.
#speedmph и *#speedkph* Эти параметры определяют максимальную скорость поезда при заезде на поворотный круг в милях/ч или км/ч. Эта скорость также будет применяться к стойловым путям.
При выходе с поворотного круга на пути доступа, поезд автоматически вернется к максимальной скорости, которая применялась при подходе к поворотному кругу.
За счёт этих команд, нет необходимости размещать указатели скорости на путях, чтобы ограничить скорость на поворотном круге.
#framerate Этот параметр определяет частоту кадров анимации при вращении поворотного круга. Дополнительные сведения см. в разделе Скорость вращения поворотного круга.
#approachclearance Этот параметр задает расстояние в метрах, на котором локомотив остановится перед поворотным кругом, чтобы дождаться его установки. Это также расстояние, на котором применяется ограничение скорости движения по поворотному кругу.
#releaseclearance Этот параметр задает расстояние в метрах, на котором поворотный круг будет освобожден после того, как локомотив с него уйдет. Это, также, расстояние, на котором отменяется ограничение скорости движения по поворотному кругу, когда локомотив покидает пул.
#settings Эквивалентно команде пула без поворотного круга с тем же именем.

Использование поворотного круга

Ни в коем случае не перемещайте поворотный круг с помощью ручного управления.

Когда поезд игрока будет извлекаться из пула, поворотный круг сам повернется в требуемое положение. При этом можно либо ждать, либо медленно двигаться к кругу. Когда Ваш поезд приближается к поворотному кругу по пути доступа, а тот не находится в нужном положении, остановитесь рядом и ждите его установки: по готовности, на экране появится уведомление.
При заезде на поворотный круг продолжайте движение до тех пор, пока локомотив не будет полностью там размещён. При правильной установке локомотива появится уведомление на экране.
После этого, установите контроллер на 0%, а реверсор - в нейтральное положение (или 0% для паровозов). Поворотный круг начнет двигаться, когда будут выполнены оба условия. Не двигайте локомотив во время вращения круга.
Когда поворотный круг займёт нужное положение, можно съехать с него.

Поведение поворотного круга, управляемого компьютером

Поворотный круг всегда будет перемещаться в требуемое положение кратчайшим путём.
Когда поезд запрашивает разворот, но круг уже активирован или занят другим поездом - запрос ставится в очередь. Поворотный круг освобождается, когда локомотив уйдет с него и удалится на небольшое расстояние. Если отсутствуют другие запросы, он останется в таком положении до следующего запроса.
Когда поезд управляемый компьютером, приближается по пути доступа к не находящемуся в требуемом положении поворотному кругу, он остановится рядом и запросит поворот круга в нужном положении.
Когда поезд управляемый компьютером подаётся из пула, а поворотный круг не находится в нужном положении, первый запросит поворот круга, но не начнет двигаться, пока второй не будет установлен.

Маршруты через поворотный круг

Программа просмотра маршрутов показывает пути, идущие через поворотный круг. Однако маршруты пула веерного типа должны начинаться за пределами области поворотного круга, и не проходить через него - как показано на этом рисунке:
Желательно иметь отдельные пути доступа для извлечения поездов из пула и отправки

поездов в пул, особенно если поворотный круг совместно используется несколькими пулами. В противном случае, если поезд отправляется на поворотный круг примерно в то же время, когда извлекается другой, существует риск возникновения тупиковой ситуации. Программа не может решить эту проблему, так как она ещё не видит, что оба поезда будут следовать по одному и тому же пути, пока извлекаемый поезд ждет у поворотного круга или поворачивается.

Скорость вращения поворотного круга

Обычно частота кадров анимации поворотного круга (скорость, с которой тот вращается) берется из файла его модели. Однако, поскольку поезда, управляемые компьютером, могут использовать поворотный круг в любом месте полигона, не исключено, что файл модели поворотного круга, не находящегося в поле зрения, не был загружен, и частота кадров не может быть получена оттуда. Значение, определенное для пула, используется в качестве замены. **Параметр опционален**: когда не задан, используется значение по умолчанию - 30 кадров в секунду, что дает скорость вращения 3 градуса в секунду. Как только файл модели используемого поворотного круга будет загружен - это значение заменится заданным в файле модели. Один кадр в секунду соответствует скорости вращения 0,1 градуса в секунду.

11.8 Изменение погоды

Изменения облачного покрова, осадков и видимости в течение суток, можно запрограммировать с помощью файлов погоды. Выбирая их в главном меню, когда активен режим «*Расписание*», игрок может получать динамически изменяющуюся в течение дня расписания погоду.

Файлы погоды помещаются в специальную подпапку *WeatherFiles* корневой директории полигона и имеют расширение `*.weather-or`. Это файлы формата JSON, состоящие из одного массива с именем "`Changes`", все элементы которого представляют собой погодные явления, активируемые в определенное время. Каждый элемент представляет собой объект JSON, свойство "`Type`" которого определяет тип погодного явления. Формат файла погоды следующий:

```
{
	"Changes": [
		{
			"Type": "<тип>",
			"<property>": "<значение>"
		}
	]
}
```

Существует три типа явлений: `Clear`, `Precipitation` и `Fog`, каждый из которых имеет свой индивидуальный набор свойств.

11.8.1 Тип " `Clear` "

Явление «Ясность» прекращает все осадки и туман, а также задаёт текущее состояние облачности. Типу «Ясность» присущи следующие свойства JSON:

Пасмурная погода

Свойство	Тип	Описание
`Time`	*текст*	Время начала явления, (*в 24-часовом формате*)
`Overcast`	*число*	Интенсивность облачности (*в процентах от 0 до 100*)
`OvercastVariation`	*число*	Разброс интенсивности облачности (*в процентах от 0 до 100*)
`OvercastRateOfChange`	*число*	Скорость изменения интенсивности облачности (*в виде коэффициента масштабирования, от 0 до 1*)
`OvercastVisibility`	*число*	Результирующая видимость (*в метрах*) *(Значение должно быть в диапазоне от 10000 до 60000)* (Для более низких значений используйте явление «туман»)

11.8.2 Тип " *Precipitation* "

Явление «Осадки» представляет собой период дождя/снегопада и последующее прояснение, с плавными переходами к началу, к концу и между фазами.

Свойство	Тип	Описание
Time	*текст*	Время начала явления, (*в 24-часовом формате*)
Фаза 1: **Нарастание облачности**		
OvercastPrecipitationStart	*число*	Интенсивность облачности при нарастании до начала осадков (*в процентах от 0 до 100*)
OvercastBuildUp	*число*	Скорость изменения интенсивности (нарастания) облачности в преддверии выпадения осадков (*в виде коэффициента масштабирования, от 0 до 1*)
PrecipitationStartPhase	*число*	Продолжительность фазы нарастания облачности в секундах (*Значение должно быть в диапазоне от 30 до 240*)
Фаза 2: **Период осадков**		
PrecipitationType	*текст*	Тип осадков. Должно быть или *Snow*, или *Rain*
PrecipitationDensity	*число*	Интенсивность осадков (*в виде коэффициента масштабирования от 0 до 1*)
PrecipitationVariation	*число*	Изменчивость интенсивности осадков (*в виде коэффициента масштабирования от 0 до 1*)
PrecipitationProbability	*число*	Вероятность выпадения осадков (*в процентах от 0 до 100*)
PrecipitationSpread	*число*	Количество отдельных зарядов во время периода осадков (*Значение должно быть, в диапазоне от 1 до 1000*)
PrecipitationVisibilityAtMinDensity	*число*	Видимость при минимальной плотности осадков
PrecipitationVisibilityAtMaxDensity	*число*	Видимость при максимальной плотности осадков
Фаза 3: **Рассеяние облачности**		
OvercastDispersion	*число*	Скорость изменения интенсивности (рассеяния) облачности по окончании периода осадков (*в виде коэффициента масштабирования от 0 до 1*)
PrecipitationEndPhase	*число*	Продолжительность фазы рассеяния облачности, в секундах. (*Значение должно быть в диапазоне от 30 до 360*)
Фаза 4: **Период прояснения**		
Overcast	*число*	Интенсивность облачности (*в процентах от 0 до 100*)
OvercastVariation	*число*	Изменение интенсивности облачности (*в процентах от 0 до 100*)
OvercastRateOfChange	*число*	Скорость изменения интенсивности облачности (*в виде коэффициента масштабирования от 0 до 1*)
OvercastVisibility	*число*	Результирующая видимость (*в метрах*) *(Значение должно быть в диапазоне от 10000 до 60000)*

11.8.3 Тип "*Fog*"

Явление «Туман» значительно снижает текущую видимость. Происходят плавные переходы к туману и от него; от предыдущего погодного явления к следующему.

Свойство	Тип	Описание
Time	*текст*	Время начала явления (*в 24-часовом формате*)
FogVisibility	*число*	Целевая видимость (*в метрах, максимальное значение 1000)* (Для более высоких значений используйте явление «Ясность»)
FogSetTime	*число*	Время перехода к туману (*в секундах*) *(Значение должно быть в диапазоне от 300 до 3600)*
FogLiftTime	*число*	Время перехода до того, как туман рассеется (*в секундах*). *Значение должно находиться в диапазоне от 360 до 3600*.
FogOvercast	*число*	Результирующая интенсивность облачности после того, как туман рассеется (в *процентах от 0 до 100*)