CodeRacing 2015

Правила

Версия 1.0.3

Ноябрь — декабрь, 2015

Оглавление

Глава 1

Объявление о проведении Конкурса

Общество с ограниченной ответственностью «Мэйл.Ру», созданное и действующее в соответствии с законодательством Российской Федерации, с местом нахождения по адресу: 125167, г. Москва, Ленинградский проспект, д. 39, строение 79, далее по тексту «Организатор конкурса», приглашает физических лиц, достигших к моменту опубликования настоящего Объявления о конкурсе 18 лет, далее по тексту «Участник конкурса», к участию в конкурсе на нижеследующих условиях:

1.1 Наименование Конкурса

«Российский кубок по программированию искусственного интеллекта (Russian AI Cup)».

Целями проведения Конкурса являются:

- повышение общественного интереса к сфере создания программных продуктов;
- предоставление Участникам конкурса возможности раскрыть творческие способности;
- развитие профессиональных навыков Участников конкурса.

Конкурс состоит из 3 (трёх) этапов, каждый из которых завершается определением Победителей. Последний этап Конкурса является решающим для Участников конкурса в состязании за получение звания Победителя Конкурса, занявшего соответствующее призовое место.

1.2 Информация об организаторе конкурса

Наименование: ООО «Мэйл.Ру»

Адрес места нахождения: 125167, г. Москва, Ленинградский проспект, д. 39, строение 79

Почтовый адрес: 125167, г. Москва, Ленинградский проспект, д. 39, строение 79, БЦ «SkyLight»

Телефон: (495) 725-63-57

Сайт: http://www.russianaicup.ru

E-мейл: russianaicup@corp.mail.ru

1.3 Сроки проведения Конкурса

Срок проведения Конкурса: с 00.00 часов 9 ноября 2015 года до 24.00 часов 20 декабря 2015 года по Московскому времени.

Первая неделя Конкурса (с 00.00 часов 9 ноября 2015 года до 24.00 часов 15 ноября 2015 года) является тестовой. В течение этого периода функциональность сайта и тестирующей системы Конкурса может быть неполной, а в правила могут вноситься существенные изменения.

Сроки начала и окончания этапов Конкурса:

- первый этап — с 00 часов 00 минут 28 ноября 2015 года до 24 часов 00 минут 29 ноября 2015 года;
- второй этап — с 00 часов 00 минут 5 декабря 2015 года до 24 часов 00 минут 6 декабря 2015 года;
- третий этап (заключительный) — с 00 часов 00 минут 12 декабря 2015 года до 24 часов 00 минут 13 декабря 2015 года.

1.4 Условие получения статуса Участника конкурса

Для участия в Конкурсе необходимо пройти процедуру регистрации в Системе Организатора конкурса, размещённой на сайте Организатора конкурса в сети Интернет по адресу: http://www.russianaicup.ru.

1.5 Срок регистрации Участников конкурса в Системе Организатора

Регистрация Участников конкурса проводится с 00.00 часов 9 ноября 2015 года до 24.00 часов 20 декабря 2015 года включительно.

1.6 Территория проведения Конкурса

Конкурс проводится на территории Российской Федерации. Проведение всех этапов Конкурса осуществляется путем удалённого доступа к Системе Организатора конкурса через сеть Интернет.

1.7 Условия проведения Конкурса (существо заданий, критерии и порядок оценки)

Порядок проведения Конкурса, существо задания, критерии и порядок оценки указаны в конкурсной документации в разделе 2.1.

Конкурсная документация включает в себя:

- Объявление о проведении Конкурса;
- Соглашение об организации и порядке проведения Конкурса;
- Правила проведения Конкурса;

- информационные данные, содержащиеся в Системе Организатора конкурса.

Участник конкурса может ознакомиться с конкурсной документацией на сайте Организатора конкурса в сети Интернет по адресу: http://www.russianaicup.ru, а также при прохождении процедуры регистрации в Системе Организатора конкурса.

Организатор конкурса оставляет за собой право на изменение конкурсной документации, условий проведения Конкурса и отказ от его проведения в соответствии с условиями конкурсной документации и нормами законодательства РФ. При этом, Организатор Конкурса обязуется уведомить Участников конкурса обо всех произошедших изменениях путём отправки уведомления, в порядке и на условиях, предусмотренных в конкурсной документации.

1.8 Порядок определения Победителей и вручения Призов. Призовой фонд Конкурса

Критерии оценки результатов Конкурса, количество и порядок определения Победителей содержатся в разделе 2.1 данного документа.

Призовой фонд Конкурса формируется за счет средств Организатора конкурса.

Призовой фонд:

- 1 место — Apple Macbook Pro 13";
- 2-3 места — Apple Macbook Air 13";
- 4-6 места — Apple iPad Air 2;
- 1-3 места в квалификации (Песочница) — WD My Passport Ultra 500 GB;
- 4-6 места в квалификации (Песочница) — Kingston SSD Now V300 120GB.

Все участники Конкурса, принявшие участие во втором или третьем этапах, будут награждены футболкой. Все участники Конкурса, принявшие участие в третьем этапе, также получат толстовку с символикой соревнования.

Все участники, занявшие призовые места, будут оповещены посредством отправки сообщения на адрес электронной почты, указанный участником при регистрации в Системе Организатора.

Призы будут высланы участникам в виде посылок, используя Почту России или другую почтовую службу, в течение двух месяцев после окончания финального этапа. Срок доставки приза по почтовому адресу, указанному участником, зависит от сроков доставки используемой почтовой службы. Почтовые адреса призёров для отправки призов Организатор получает из учётных данных участника в Системе Организатора. Адрес должен быть указан участником-призёром в течение трёх дней после получения уведомления о получении приза.

При отсутствии ответа в обозначенные сроки или отказе предоставить точные данные, необходимые для вручения призов Конкурса, Организатор оставляет за собой право отказать такому участнику в выдаче приза Конкурса. Денежный эквивалент приза не выдаётся.

Победители Конкурса обязуются предоставить Организатору конкурса копии всех документов, необходимых для бухгалтерской и налоговой отчётности Организатора конкурса. Перечень документов, которые Победитель обязан предоставить Организатору конкурса, может включать в себя:

- копию паспорта Победителя;

- копию свидетельства о постановке на налоговый учет Победителя;
- копию пенсионного удостоверения Победителя;
- данные об открытии банковского лицевого счета Победителя;
- иные документы, которые Организатор конкурса потребует от Участника конкурса в целях формирования отчётности о проведённом Конкурсе.

Наряду с копиями Организатор конкурса вправе запросить оригиналы вышеуказанных документов.

В соответствии с подпунктом 4 пункта 1 статьи 228 НК РФ Победитель Конкурса, ставший обладателем Приза, самостоятельно несёт все расходы по уплате всех применимых налогов, установленных действующим законодательством Российской Федерации.

1.9 Порядок и способ информирования участников Конкурса

Информирование Участников Конкурса осуществляется путём размещения информации в сети Интернет на Сайте Организатора конкурса по адресу: http://www.russianaicup.ru, а также через Систему Организатора конкурса, в течение всего срока проведения Конкурса.

Глава 2

О мире CodeRacing 2015

2.1 Общие положения игры и правила проведения турнира

Данное соревнование предоставляет вам возможность проверить свои навыки программирования, создав искусственный интеллект (стратегию), управляющий одним или группой из двух кодемобилей в специальном игровом мире (подробнее об особенностях мира CodeRacing 2015 можно узнать в следующих разделах). В каждой игре вам будет противостоять одна или несколько стратегий других игроков. Основной целью каждого кодемобиля является максимально быстрое прохождение 2 кругов замкнутой трассы. За каждый пройденный круг и дополнительно, после завершения второго, кодемобиль зарабатывает баллы, необходимые вам для победы. Однако даже в гонках «Формулы-1» иногда применяются не совсем честные действия, не оптимальные для одиночного прохождения трассы, но мешающие соперникам в групповом заезде добраться до финиша раньше вас. В мире кодемобилей подобные «грязные» приёмы не просто являются нормой, но и разрешены официально. Вы можете мешать вашему сопернику, наносить ему повреждения различными способами и даже временно выводить из строя его кодемобиль, получая за это дополнительные баллы. Звание победителя игры, а также все остальные места распределяются в соответствии с количеством набранных баллов. Два или более игроков могут делить одно место, если их баллы равны.

Турнир проводится в несколько этапов, которым предшествует квалификация в Песочнице. Песочница — соревнование, которое проходит на протяжении всего чемпионата. В рамках каждого этапа игроку соответствует некоторое значение рейтинга — показателя того, насколько успешно его стратегия участвует в играх.

Начальное значение рейтинга в Песочнице равно 1200. По итогам игры это значение может как увеличиться, так и уменьшиться. При этом победа над слабым (с низким рейтингом) противником даёт небольшой прирост, также и поражение от сильного соперника незначительно уменьшает ваш рейтинг. Со временем рейтинг в Песочнице становится всё более инертным, что позволяет уменьшить влияние случайных длинных серий побед или поражений на место участника, однако вместе с тем и затрудняет изменение его положения при существенном улучшении стратегии. Для отмены данного эффекта участник может сбросить изменчивость рейтинга до начального состояния при отправке новой стратегии, включив соответствующую опцию. В случае принятия новой стратегии системой рейтинг участника мгновенно упадёт, однако по мере участия в играх быстро восстановится и даже станет выше, если ваша стратегия действительно стала эффективнее. Не рекомендуется использовать данную опцию при незначительных, инкрементальных улучшениях вашей стратегии, а также в случаях, когда новая стратегия недостаточно протестирована и эффект от изменений в ней достоверно не известен.

Начальное значение рейтинга на каждом основном этапе турнира равно 0. За каждую игру участник получает определённое количество единиц рейтинга в зависимости от занятого в бою места (система, аналогичная используемой в чемпионате «Формула-1»). Если двое или более участников делят какое-то место, то суммарное количество единиц рейтинга за это место и за следующие

`количество_таких_участников` − 1 мест делится поровну между этими участниками. Например, если двое участников делят третье место, то каждый из них получит половину от суммы единиц рейтинга за третье и четвёртое места. При делении округление всегда совершается в меньшую сторону. Более подробная информация об этапах турнира будет представлена в анонсах на сайте проекта.

Сначала все участники могут участвовать только в играх, проходящих в Песочнице. Игроки могут отправлять в Песочницу свои стратегии, и последняя принятая из них берётся системой для участия в квалификационных играх. Каждый игрок участвует примерно в одной квалификационной игре за час. Жюри оставляет за собой право изменить этот интервал, исходя из пропускной способности тестирующей системы, однако для всех участников он остаётся постоянной величиной. Игры в Песочнице проходят по правилам, соответствующим правилам случайного прошедшего этапа турнира или же правилам следующего (текущего) этапа. При этом, чем ближе значение рейтинга двух игроков в рамках Песочницы, тем больше вероятность того, что они окажутся в одной игре. Песочница стартует до начала первого этапа турнира и завершается через некоторое время после финального (смотрите расписание этапов для уточнения подробностей). Помимо этого Песочница замораживается на время проведения этапов турнира. По итогам игр в Песочнице происходит отбор для участия в Раунде 1, в который пройдут 900 участников с наибольшим рейтингом (при его равенстве приоритет отдаётся игроку, раньше отправившему последнюю версию своей стратегии).

Этапы турнира:

- В Раунде 1 вам предстоит освоить базовое управление кодемобилем на небольшом наборе гоночных трасс, а также изучить некоторые особенности кодемобиля багги. В каждой игре данного этапа примет участие 4 игрока, у которых будет по одному кодемобилю указанного типа. Этот этап, как и все последующие, состоит из двух частей, между которыми будет небольшой перерыв (с возобновлением работы Песочницы), который позволит улучшить свою стратегию. Для игр в каждой части выбирается последняя стратегия, отправленная игроком до начала этой части. Игры проводятся волнами. В каждой волне каждый игрок участвует ровно в одной игре. Количество волн в каждой части определяется возможностями тестирующей системы, но гарантируется, что оно не будет меньше десяти. 300 участников с наиболее высоким рейтингом пройдут в Раунд 2. Также в Раунд 2 будет проведён добор 60 участников с наибольшим рейтингом в Песочнице (на момент начала Раунда 2) из числа тех, кто не прошёл по итогам Раунда 1.
- В Раунде 2 вам предстоит улучшить свои навыки управления кодемобилем, освоить расширенный набор трасс, а также изучить некоторые особенности другого кодемобиля — джипа. Так же, как и в предыдущем этапе, игры будут проходить в формате 4×1 — 4 игрока, по одному кодемобилю у каждого. Между этапами будет некоторый перерыв, позволяющий доработать стратегию. Усложняет задачу то, что после подведения итогов Раунда 1 часть слабых стратегий будет отсеяна и вам придётся противостоять более сильным соперникам. По итогам Раунда 2 лучшие 50 стратегий попадут в Финал. Также в Финал будет проведен добор 10 участников с наибольшим рейтингом в Песочнице (на момент начала Финала) из числа тех, кто не прошёл в рамках основного турнира.
- Финал является самым серьёзным этапом. После отбора, проведённого по итогам двух первых этапов, останутся сильнейшие. И в каждой игре вам придётся сойтись лицом к лицу с одним из них. Для победы вам необходимо не только обобщить навыки управления различными кодемобилями, полученные на предыдущих этапах соревнования, но также и реализовать координацию действий ваших кодемобилей. Только слаженная работа приведёт вашу команду к желаемому результату. Дополнительную сложность представляет то, что трассы Финала не будут известны заранее, а стратегиям участников придётся принимать решения в условиях частичной видимости. Игры Финала будут проходить в формате 2×2 — 2 игрока, по одному кодемобилю каждого типа у каждого игрока. Система проведения Финала имеет свои особенности. Этап по-прежнему делится на две части, однако они уже не будут состоять из волн. Для каждой пары участников Финала будет проведено две игры. Гоночные трассы не являются симметричными. Поэтому в целях уменьшения влияния начальной позиции кодемобилей на результат игры вторая из этих игр будет отличаться от первой лишь тем, что кодемобили участников поменяются местами (багги первого участника поменяется местом с багги второго, соответственно и джипы обоих участников поменяются местами между собой). Если позволит время и возможности тестирующей системы, описанная серия игр будет повторена.

После окончания Финала все финалисты упорядочиваются по невозрастанию рейтинга. При равенстве

рейтингов более высокое место занимает тот финалист, чья участвовавшая в финале стратегия была отослана раньше. Призы за Финал распределяются на основании занятого места после этого упорядочивания. Лучшие шесть финалистов награждаются призами:

- 1 место — Apple Macbook Pro 13";
- 2-3 места — Apple Macbook Air 13";
- 4-6 места — Apple iPad Air 2;

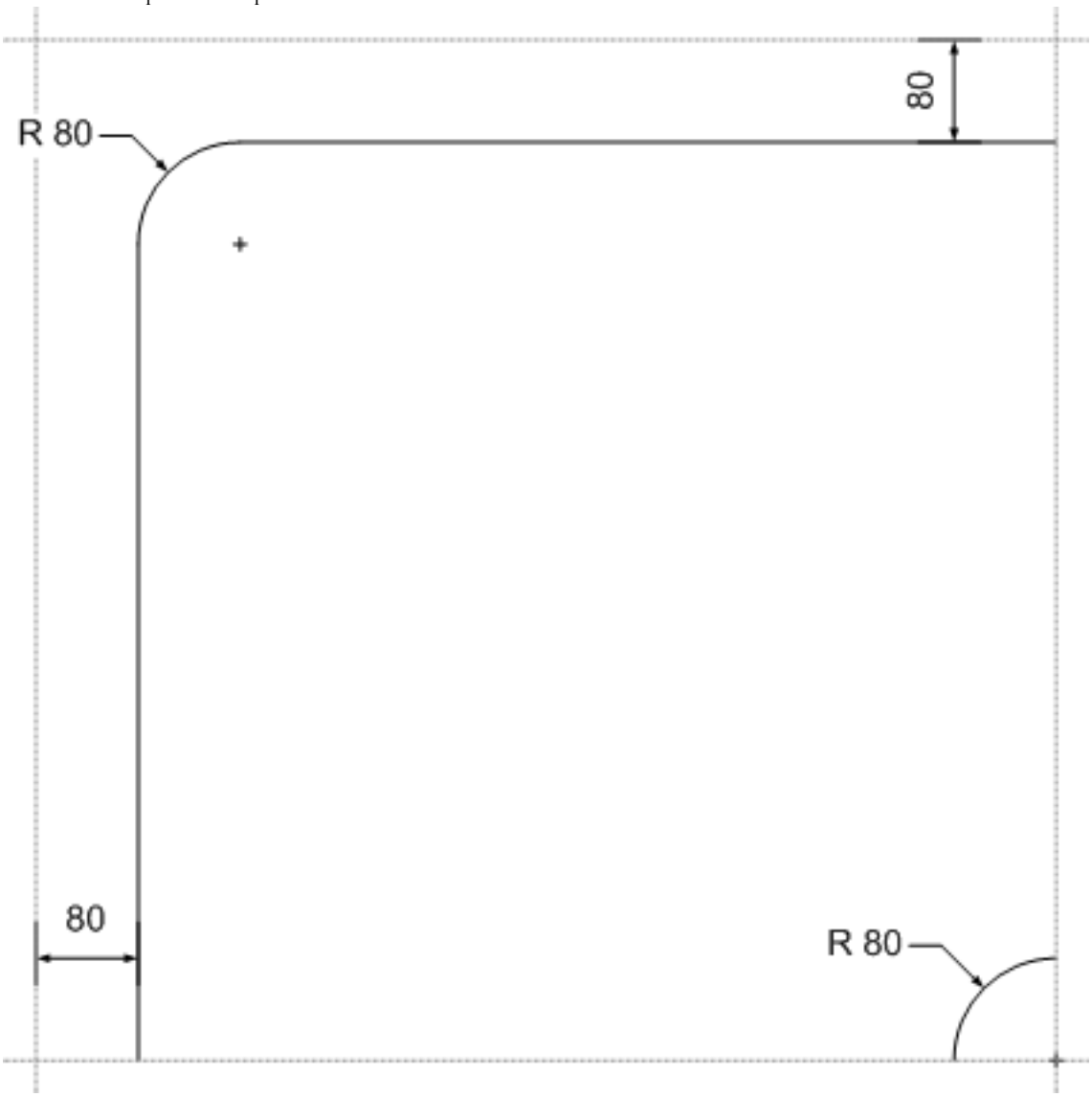
После окончания Песочницы все её участники, кроме призёров Финала, упорядочиваются по невозрастанию рейтинга. При равенстве рейтингов более высокое место занимает тот участник, который раньше отослал последнюю версию своей стратегии. Призы за Песочницу распределяются на основании занятого места после этого упорядочивания. Лучшие шесть участников Песочницы награждаются ценными подарками.

2.2 Описание игрового мира

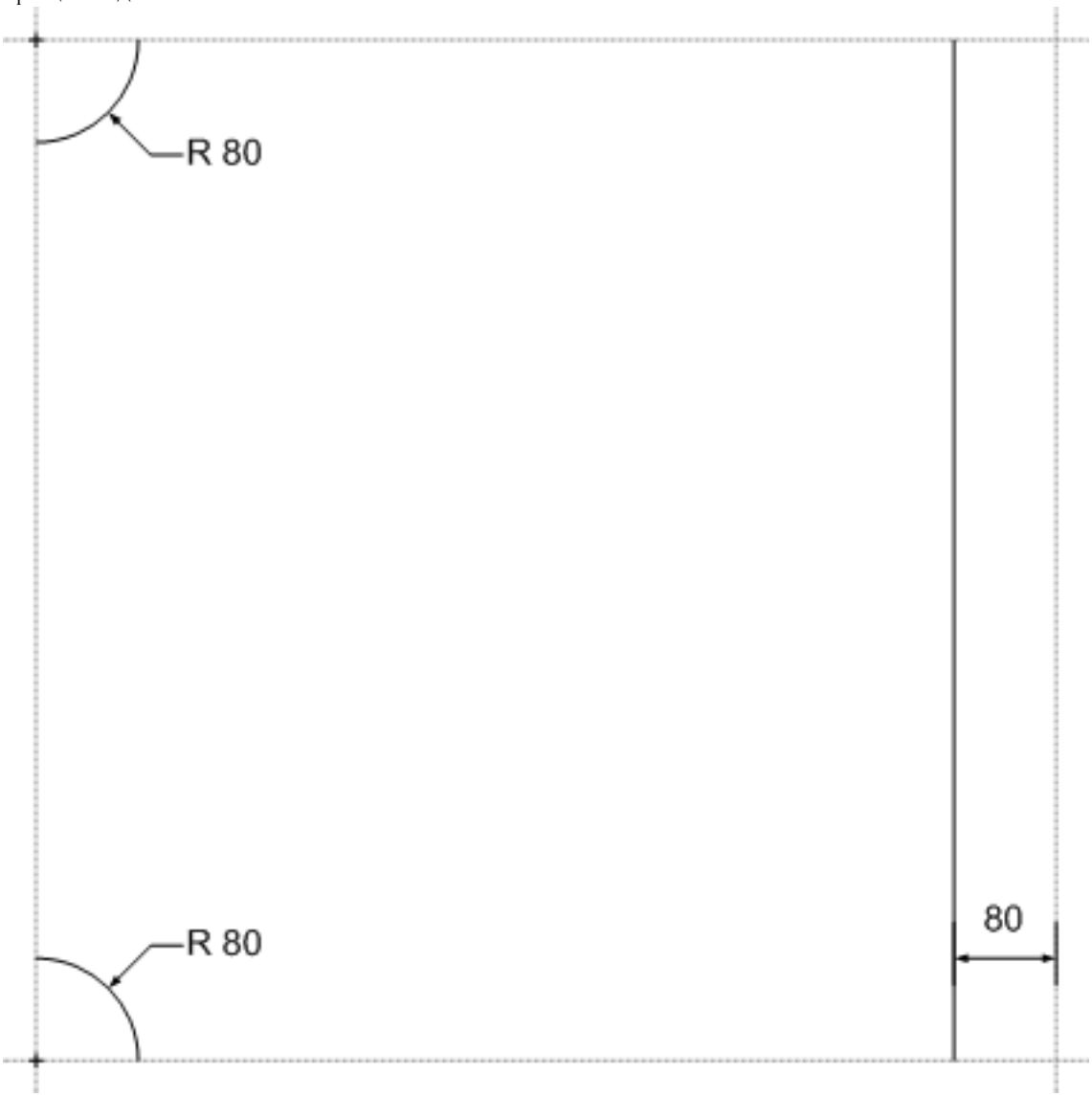
Игровой мир является двумерным. Ось абсцисс в этом мире направлена слева направо, ось ординат — сверху вниз, угол 0.0 совпадает с направлением оси абсцисс, а положительный угол вращения означает вращение по часовой стрелке. Гоночная трасса может быть представлена в виде прямоугольной матрицы, элементами которой являются «тайлы». Каждое из измерений матрицы (количество строк и количество столбцов) находится в интервале от 8 до 16 включительно. Тайл — это квадратная область размером 800×800. В игре есть 12 типов тайлов: 4 поворота (например, `LEFT_TOP_CORNER`), 4 Т-образных перекрёстка (например, `LEFT_HEADED_T`), прямой вертикальный участок дороги (`VERTICAL`), прямой горизонтальный участок дороги (`HORIZONTAL`), перекрёсток (`CROSSROADS`), а также тайл, не содержащий участка дороги (`EMPTY`). Отступ от границы тайла до прямого участка трассы, проходящего через этот тайл, равен 80. Радиус всех закруглений участков трассы также равен 80. Все тайлы одного типа полностью идентичны друг другу.

Далее приведены схемы некоторых тайлов:

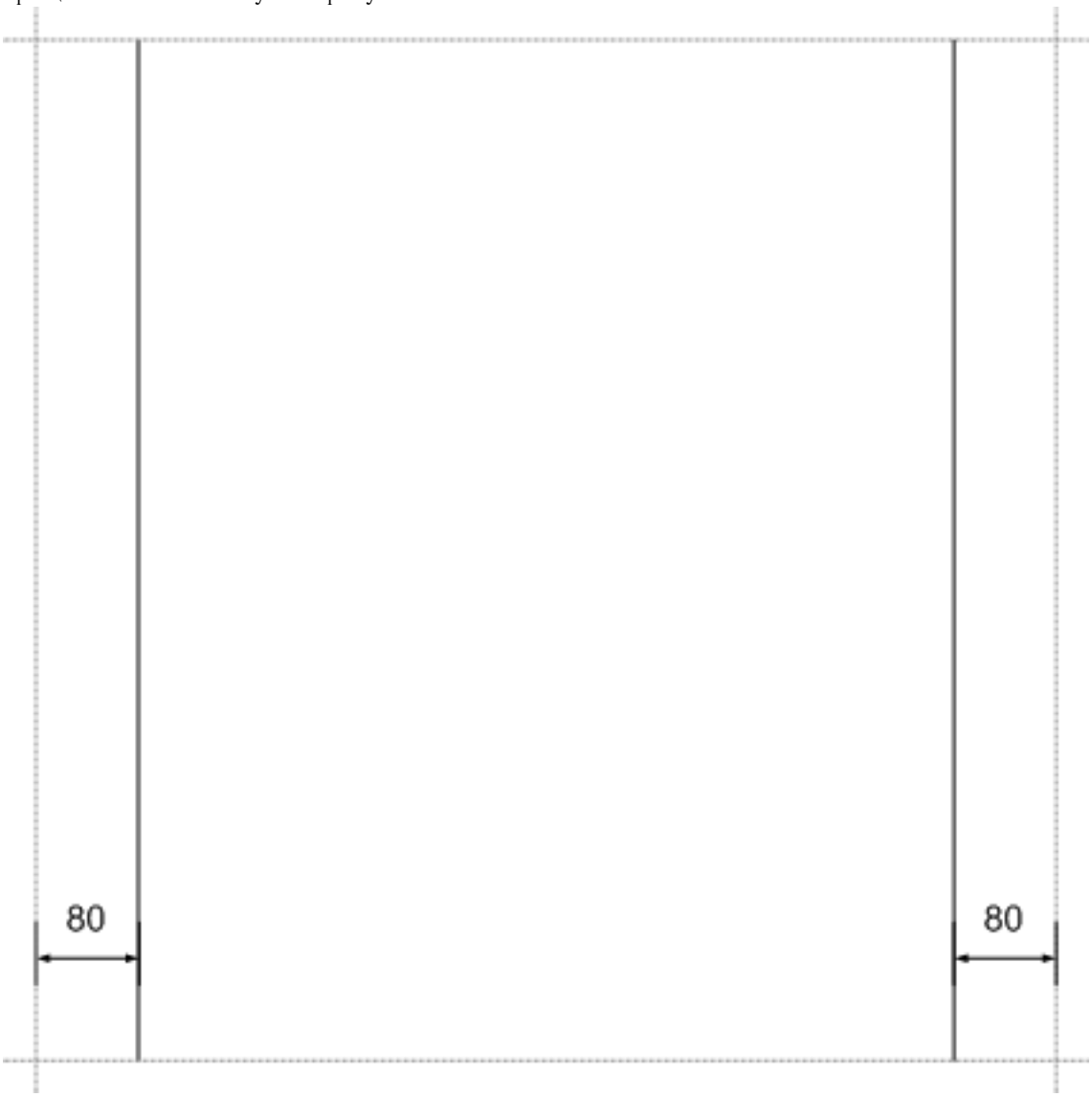
- Схема тайла типа `LEFT_TOP_CORNER`, служащего для соединения двух других непустых тайлов — справа и снизу от данного. Другие три поворота могут быть получены путём вращения данного тайла в плоскости игрового мира с шагом 90°.



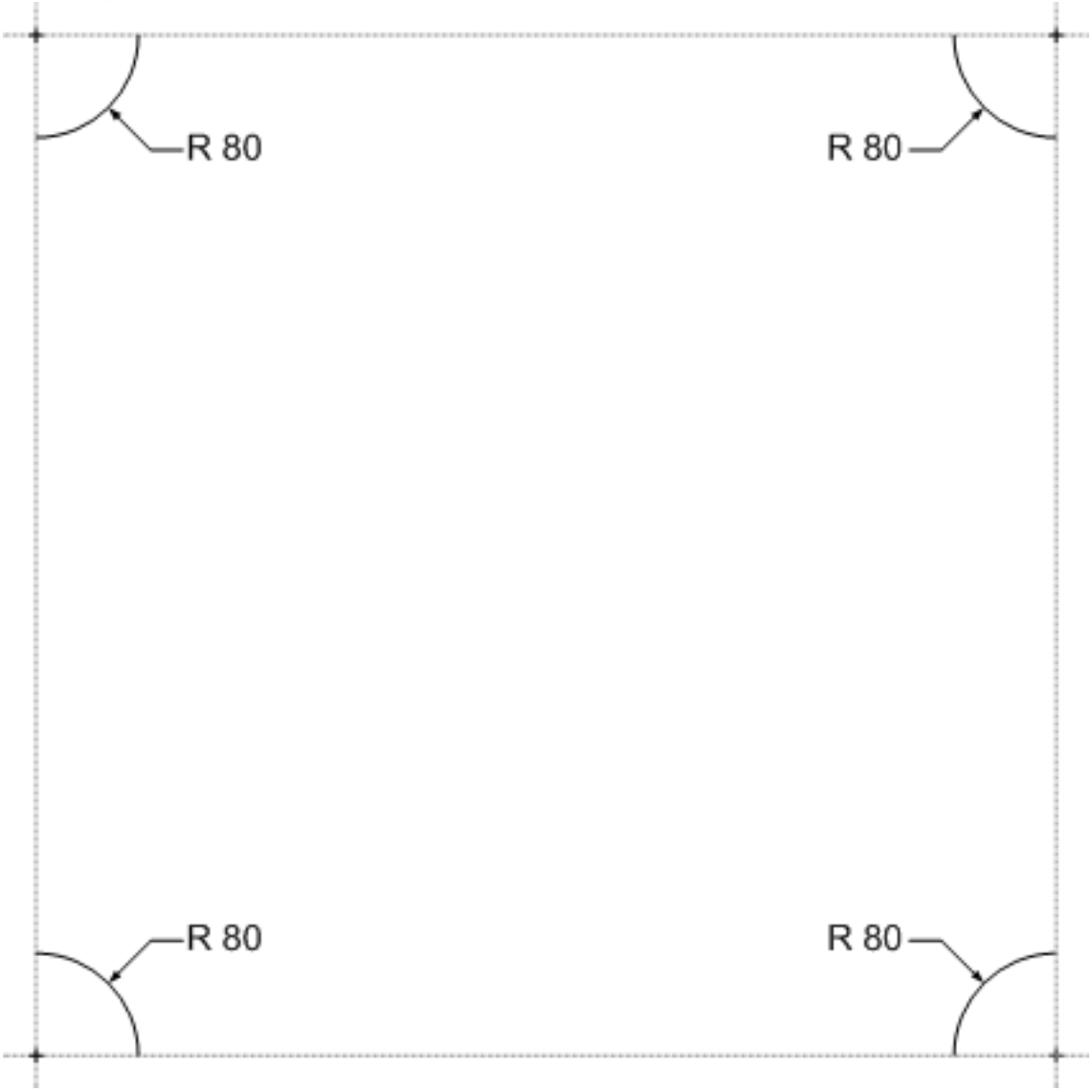
- Схема тайла типа `LEFT_HEADED_T`, служащего для соединения трёх других непустых тайлов — слева, сверху и снизу от данного. Другие три T-образных участка дороги могут быть получены путём вращения данного тайла с шагом 90°.



- Схема вертикального участка дороги, горизонтальный участок может быть получен из него путём вращения на 90° в любую сторону.



- Схема перекрёстка.



Время в игре дискретное и измеряется в «тиках». В начале каждого тика игра получает от стратегий желаемые действия кодемобилей в этот тик и обновляет состояние кодемобилей в соответствии с этими желаниями и ограничениями мира. Затем происходит расчёт изменения мира и объектов в нём за этот тик, и процесс повторяется снова с обновлёнными данными. Базовая длительность каждой игры определяется эвристическим алгоритмом и примерно пропорциональна длине и сложности гоночной трассы. Как правило, она находится в интервале от 5000 до 15000 тиков. К полученному значению прибавляется 180 — количество тиков в начале игры, в течение которых стратегия получает управление кодемобилем, однако не может изменять его положение и скорость. Если какой-либо кодемобиль финиширует (завершает второй круг трассы), то для остальных кодемобилей с текущего момента устанавливается дедлайн, равный 20% базовой длительности игры и округлённый вниз до целого количества тиков. Если в течение этого времени ни один кодемобиль не придёт к финишу, игра завершается преждевременно. В противном случае после каждого финишировавшего кодемобиля будет установлен новый дедлайн. Установка дедлайна не может продлить игру свыше её базовой длительности. Если для каждого кодемобиля верно, что он финишировал трассу либо стратегия, управляющая им, «упала», игра завершается преждевременно.

«Упавшая» стратегия больше не может управлять кодемобилями. Стратегия считается «упавшей» в следующих случаях:

- Процесс, в котором запущена стратегия, непредвиденно завершился, либо произошла ошибка в

протоколе взаимодействия между стратегией и игровым сервером.

- Стратегия превысила одно (любое) из отведённых ей ограничений по времени. Стратегии на один тик выделяется не более 5 секунд реального времени. Но в сумме на всю игру процессу стратегии выделяется

  $$30 \times <длительность\_игры\_в\_тиках> \times <количество\_кодемобилей\_в\_команде> + 5000 \quad (2.1)$$

  миллисекунд реального времени и

  $$15 \times <длительность\_игры\_в\_тиках> \times <количество\_кодемобилей\_в\_команде> + 5000 \quad (2.2)$$

  миллисекунд процессорного времени. [1] В формуле учитывается только базовая длительность игры. Ограничение по времени остаётся прежним, даже если реальная длительность игры отличается от этого значения. Все ограничения по времени распространяются не только на код участника, но и на взаимодействие клиента-оболочки стратегии с игровым симулятором.

- Стратегия превысила ограничение по памяти. В любой момент времени процесс стратегии не должен потреблять более 256 Мб оперативной памяти.

Гонки Финала будут проходить в режиме частичной видимости. По умолчанию в каждой ячейке матрицы тайлов гоночной трассы будет находиться специальное значение `UNKNOWN`, показывающее, что тип тайла вам (пока) не известен. Тайлы будут открываться по мере прохождения трассы, и в большинстве случаев вся трасса станет известна вам после завершения первого круга. В каждый тик кодемобиль открывает тайл, в котором он непосредственно находится, а также все тайлы, манхэттенское расстояние которых от данного тайла не превышает 2 — всего до 13 тайлов. Стратегия участника будет получать данные обо всех юнитах, находящихся в открытых тайлах, но не о юнитах в тайлах со значением `UNKNOWN`.

Физика игрового мира основана на движке Notreal2D, специально разработанном для CodeRacing 2015 и других проектов серии Russian AI Cup. Исходный код Notreal2D опубликован на Github: https://github.com/Russian-AI-Cup/notreal2d.

2.3 Описание типов юнитов

В мире CodeRacing 2015 существует 4 типа юнитов: кодемобили, снаряды, бонусы и лужи мазута. В свою очередь существует два типа кодемобилей: багги (`BUGGY`) и джип (`JEEP`); два типа снарядов: шайба (`WASHER`) и шина (`TIRE`); а также пять типов бонусов: ремкомплект (`REPAIR_KIT`), ящик со снарядами (`AMMO_CRATE`), заряд для системы закиси азота (`NITRO_BOOST`), канистра с мазутом (`OIL_CANISTER`) и дополнительные баллы (`PURE_SCORE`). Сравнительные массогабаритные характеристики юнитов приведены в следующей таблице:

Характеристика юнита	Багги	Джип	Шайба	Шина	Бонус	Лужа мазута
Форма	прямоуг.	прямоуг.	круг	круг	прямоуг.	круг
Ширина/диаметр	210	210	40	140	70	300
Высота	140	140	—	—	70	—
Масса	1250	1500	10	1000	100	—

2.4 Характеристики и управление кодемобилем

Основной характеристикой кодемобиля является прочность. Начальная (и максимальная) прочность каждого кодемобиля равна 1.0. Если прочность падает до нуля, кодемобиль считается выведенным из строя

[1] Несмотря на то, что ограничение реального времени заметно выше ограничения процессорного времени, запрещено искусственно «замедлять» тестирование стратегии командами типа «`sleep`» (равно как и пытаться замедлить/дестабилизировать тестирующую систему другими способами). В случае выявления подобных злоупотреблений, жюри оставляет за собой право применить к данному пользователю меры на своё усмотрение, вплоть до дисквалификации из соревнования и блокировки аккаунта.

и перестаёт управляться стратегией. Однако через 300 тиков кодемобиль восстанавливается (его прочность становится равной 1.0) и может продолжать движение.

Скорость и направление движения кодемобиля определяются следующими тремя параметрами:

- Относительная мощность работы двигателя `car.enginePower`. Значение находится в интервале от -1.0 до 1.0 кроме случаев, когда кодемобиль использует ускорение «нитро». При переходе кодемобиля в режим ускорения относительная мощность его двигателя мгновенно становится равной 2.0 и не меняется до окончания действия ускорения, затем она устанавливается в 1.0. Абсолютная мощность двигателя равномерно изменяется на интервале (относительной мощности) от $-\infty$ до 0.0 и на интервале от 0.0 до $+\infty$ и для каждого типа кодемобилей подобрана таким образом, что при значении `car.enginePower`, равном 1.0, ускорение кодемобиля составляет 0.25 тиков$^{-2}$ (ускорение вперёд/замедление движения назад), а при значении -1.0 — -0.1875 тиков$^{-2}$ (замедление движения вперёд/ускорение назад). В игре нет явного ограничения на скорость движения кодемобилей, однако присутствует сопротивление воздуха, которое ограничивает их скорость естественным образом. Стратегия может установить `move.enginePower` — желаемое значение относительной мощности работы, однако изменение `car.enginePower` не будет превышать по модулю 0.025 за тик (кроме входа и выхода из режима ускорения).
- Относительный поворот руля/колёс `car.wheelTurn`. Значение находится в интервале от -1.0 до 1.0 и, как и `car.enginePower`, не может изменяться мгновенно, а двигается к желаемому значению `move.wheelTurn` со скоростью, не превышающей по модулю 0.05 за тик. Ненулевой поворот колёс порождает составляющую угловой скорости кодемобиля, значение которой прямо пропорционально `car.wheelTurn`, коэффициенту `game.carAngularSpeedFactor`, а также скалярному произведению вектора скорости кодемобиля и единичного вектора, направление которого совпадает с направлением кодемобиля. Однако реальная угловая скорость может отличаться от данного значения вследствие столкновений кодемобиля с другими игровыми объектами.
- Положение педали тормоза. В любой тик стратегия может установить флажок `move.brake`. Это действие заблокирует колёса кодемобиля в данный тик и увеличит силу трения, воздействующую на кодемобиль вдоль его продольной оси, до значения силы трения, воздействующей на кодемобиль вдоль его поперечной оси. По умолчанию первая значительно меньше второй. Блокировка колёс означает, что игра будет игнорировать мощность работы двигателя кодемобиля, однако стратегия по прежнему сможет изменять это значение. Данное состояние кодемобиля схоже с состоянием в первые 180 тиков игры с тем исключением, что при нажатии педали тормоза у кодемобиля уже может быть ненулевая скорость.

У кодемобиля в распоряжении есть 3 типа расходников: метательные снаряды (их количество равно `car.projectileCount`), заряды для системы закиси азота (`car.nitroChargeCount`) и канистры с мазутом (`car.oilCanisterCount`). В начале игры у кодемобиля есть по одному расходнику каждого типа. При завершении очередного круга трассы количество расходников каждого типа у кодемобиля увеличивается на единицу. При подборе кодемобилем бонуса, содержащего расходники, количество расходников соответствующего типа увеличивается на единицу.

- Стратегия может использовать метательные снаряды для нанесения повреждений и препятствования перемещению других кодемобилей. В момент появления центр снаряда совпадает с центром кодемобиля, запустившего этот снаряд. Разумеется, кодемобиль при этом является «прозрачным» для снаряда. Игра игнорирует пересечения и столкновения кодемобиля и запущенного им снаряда до тех пор, пока снаряд не столкнётся с любым другим объектом. После этого снаряд начинает взаимодействовать со всеми кодемобилями одинаково.
  - В распоряжении багги находятся небольшие и лёгкие шайбы. Багги одновременно запускает три таких шайбы (тратится один расходник): направление первой совпадает с направлением кодемобиля, а направление двух других отличается на $+2^\circ$ и -2°. Модуль скорости каждой шайбы постоянен и равен 60 тикам$^{-1}$. Шайбы не взаимодействуют между собой и с какими-либо объектами, кроме кодемобилей и шин, а также проходят сквозь границы трассы. При попадании

в кодемобиль шайба понижает его прочность на 0.15. При столкновении с шиной шайба убирается из игрового мира.

– Джип использует в качестве снарядов огромные шины. Начальная скорость этого типа снарядов также равна 60 тикам$^{-1}$. Шины могут отскакивать друг от друга, от кодемобилей и бортов трассы, теряя при этом часть скорости. Если скорость шины падает до 25% от начальной, шина убирается из игрового мира. При столкновении с кодемобилем шина наносит ему урон, равный 35% от отношения скорости соударения[2] к начальной скорости снаряда.

После запуска очередного снаряда следующий может быть запущен не ранее, чем через 60 тиков.

- При использовании ускорения «нитро» мощность двигателя кодемобиля мгновенно становится равной 2.0 и не может быть изменена в течение 120 тиков — длительности ускорения. Разумеется, в течение этого времени кодемобиль не может совершить ещё одну активацию режима ускорения.
- Канистры с мазутом используются для создания дополнительных препятствий оппонентам, следующим за вашим кодемобилем на небольшой дистанции. При использовании расходника данного типа позади кодемобиля образуется скользкая лужа мазута. Центр лужи находится на продольной оси кодемобиля, а расстояние между ближайшими точками кодемобиля и лужи равно 10.0. Лужа мазута полностью высыхает и убирается из игрового мира через 600 тиков. Если какой-либо кодемобиль попадает в лужу мазута (расстояние от центра кодемобиля до центра лужи становится меньше её радиуса), то вследствие ухудшения сцепления колёс и поверхности дороги кодемобиль начинает «заносить», к его угловой скорости мгновенно прибавляется некоторое случайное значение, пропорциональное модулю текущей линейной скорости кодемобиля, а часть мазута попадает на колёса и остаётся там некоторое время: до 60 тиков, но не более оставшегося времени высыхания лужи. При этом, время высыхания лужи сокращается на то же значение. Пока на колёсах остаётся мазут, сила трения поверхности, воздействующая на кодемобиль, значительно уменьшается, а педаль тормоза перестаёт эффективно работать; новое попадание в лужу мазута до окончания этого срока не имеет никакого эффекта. Канистры с мазутом могут использоваться не чаще, чем раз в 120 тиков.

2.5 Столкновения юнитов

Юниты могут сталкиваться с границами трассы, а также между собой в зависимости от типов этих юнитов. При столкновении объектов модуль и направление их скоростей меняются определённым образом в зависимости от масс и исходных скоростей объектов. Столкновения не являются абсолютно упругими, и объекты теряют часть скорости.

Во время гонки кодемобили сталкиваются с границами трассы и всеми юнитами, кроме луж мазута. Финишировавшие кодемобили не убираются из игрового мира, однако перестают взаимодействовать с какими-либо объектами, кроме границ трассы.

- При столкновении двух кодемобилей оба кодемобиля могут получить некоторый урон. Урон, полученный кодемобилем, пропорционален скорости соударения и отношению массы другого кодемобиля к массе данного кодемобиля. Если урон меньше, чем 0.01, то он игнорируется.
- При столкновении с ограждением трассы кодемобиль получает урон, пропорциональный скорости соударения. Как и в случае со столкновением двух кодемобилей, урон ниже 0.01 считается несущественным и игнорируется.

---

[2] Здесь и далее под скоростью соударения юнитов А и Б понимается величина скалярного произведения вектора, полученного в результате вычитания вектора скорости юнита А в точке столкновения из вектора скорости юнита Б в точке столкновения, и нормали столкновения этих двух юнитов. Скорость юнита в произвольной точке определяется его поступательной скоростью, а также скоростью его вращения и расстоянием от центра юнита до указанной точки. Нормаль столкновения является единичным вектором, направление которого совпадает либо с нормалью к поверхности юнита Б в точке столкновения, направленной за пределы этого объекта, либо с нормалью к поверхности юнита А в точке соударения, направленной внутрь этого объекта. Определение нормали столкновения в каждом конкретном случае зависит от особенностей реализации физики игрового мира, однако для выпуклых объектов с некоторым упрощением можно считать, что направление этого вектора не сильно отличается от направления вектора, направленного из центра юнита Б в центр юнита А.

- При столкновении со снарядом кодемобиль получает урон в зависимости от типа снаряда. Подробная информация о столкновениях этого типа уже приводилась ранее.
- После столкновения кодемобиля с бонусом последний убирается из игрового мира, а кодемобиль или его владелец получают некоторое вознаграждение, в зависимости от типа бонуса:
  - Ремкомплект полностью восстанавливает прочность кодемобиля.
  - Ящик со снарядами, заряд для системы закиси азота и канистра с мазутом увеличивают количество расходников соответствующего типа в распоряжении кодемобиля на единицу.
  - Дополнительные баллы добавляют 100 к счётчику баллов игрока.
- Лужи мазута не взаимодействуют с кодемобилями и любыми другими юнитами как физические объекты. Подробная информация о воздействии лужи мазута на кодемобиль уже приводилась ранее.

Все типы столкновений, не описанные в данном документе, игнорируются симулятором игры.

2.6 Прохождение трассы и начисление баллов

Прохождение одного круга трассы заключается в последовательном посещении всех ключевых тайлов (`world.waypoints`) и возврате в начальный тайл. Тайл считается посещённым, как только центр кодемобиля пересекает границу тайла. Вероятность точного совпадения центра кодемобиля и границы тайла крайне мала, но для полного соблюдения всех формальностей стоит отметить, что тайл включает в себя левую и верхнюю границы, правая и нижняя принадлежат соседним тайлам. За полное прохождение круга кодемобиль приносит игроку ровно 1000 баллов. При этом начисление производится не единовременно, а по мере посещения ключевых тайлов. 500 баллов из указанной тысячи распределяются равномерно по всем ключевым тайлам, кроме последнего. При этом используется целочисленное деление, таким образом суммарный балл за эти тайлы может быть чуть менее 500. Кодемобиль, первым завершивший 2 круга, получает дополнительную премию размером в 512 баллов. Премия за каждое последующее финиширование трассы уменьшается вдвое.

Как уже указывалось, быстрое прохождение трассы является основным источником получения баллов. Есть и другие способы:

- Подбор бонусов, содержащих баллы. 100 баллов за каждый бонус.
- Нанесение повреждений кодемобилям других игроков. Повреждения учитываются с коэффициентом 100.0. Округление производится вниз до ближайшего целого числа.
- Выведение из строя кодемобилей других игроков. За факт выведения из строя начисляются дополнительные 100 баллов.

Количество баллов игрока равно сумме баллов, заработанных всеми его кодемобилями.

Глава 3

Создание стратегии

3.1 Техническая часть

Сперва для создания стратегии вам необходимо выбрать один из ряда поддерживаемых языков программирования[3]: Java (Oracle JDK 8), C# (Visual C# compiler 4.0.30319+ for .NET framework 4.5+), C++ (GNU MinGW C++ 5.1), Python 2 (Python 2.7+), Python 3 (Python 3.5+), Pascal (Free Pascal 2.6+), Ruby (JRuby 9.0.3.0+, Oracle JDK 8). Возможно, этот набор будет расширен. На сайте проекта вы можете скачать пользовательский пакет для каждого из языков. Модифицировать в пакете разрешено лишь один файл, который и предназначен для содержания вашей стратегии, например, MyStrategy.java (для Java) или MyStrategy.py (для Python)[4]. Все остальные файлы пакета при сборке стратегии будут замещены стандартными версиями. Однако вы можете добавлять в стратегию свои файлы с кодом. Эти файлы должны находиться в том же каталоге, что и основной файл стратегии. При отправке решения все они должны быть помещены в один ZIP-архив (файлы должны находиться в корне архива). Если вы не добавляете новых файлов в пакет, достаточно отправить сам файл стратегии (с помощью диалога выбора файла) или же вставить его код в текстовое поле.

После того, как вы отправили свою стратегию, она попадает в очередь тестирования. Система сперва попытается скомпилировать пакет с вашими файлами, а затем, если операция прошла успешно, создаст несколько коротких (по 200 тиков) игр разных форматов: 4×1 с использованием кодемобилей багги, 4×1 с использованием джипов и 2×2. Для управления кодемобилями каждого из участников этих игр будет запущен отдельный клиентский процесс с вашей стратегией, и для того, чтобы стратегия считалась принятой (корректной), ни один из экземпляров стратегии не должен «упасть». Игрокам в этих тестовых играх будут даны имена в формате «`<имя_игрока>`», «`<имя_игрока> (2)`», «`<имя_игрока> (3)`» и т.д.

После успешного прохождения описанного процесса ваша посылка получает статус «Принята». Первая успешная посылка одновременно означает и вашу регистрацию в Песочнице. Вам начисляется стартовый рейтинг (1200), и ваша стратегия начинает участвовать в периодических квалификационных боях (смотрите описание Песочницы для получения более подробной информации). Также вам становится доступна функция создания собственных игр, в которых в качестве соперника можно выбирать любую стратегию любого игрока (в том числе и вашу собственную), созданную до момента вашей последнего успешной посылки. Созданные вами игры не влияют на рейтинг.

В системе присутствуют ограничения на количество посылок и пользовательских игр, а именно:

- Нельзя отправлять стратегию чаще, чем три раза за пять минут.

---

[3] Для всех языков программирования используются 32-битные версии компиляторов/интерпретаторов.

[4] Исключение составляет C++, для которого можно модифицировать два файла: MyStrategy.cpp и MyStrategy.h. Причём наличие в архиве файла MyStrategy.cpp является обязательным (иначе стратегия не скомпилируется), а наличие файла MyStrategy.h — опциональным. В случае его отсутствия будет использован стандартный файл из пакета.

- За пять минут нельзя создать более двух пользовательских игр.

Для упрощения отладки небольших изменений стратегии в системе присутствует возможность сделать тестовую посылку (флажок «Тестовая посылка» на форме отправки стратегии). Тестовая посылка не отображается другим пользователям, не участвует в квалификационных играх в Песочнице и играх в этапах турнира, также невозможно собственноручно создавать бои с её участием. Однако после принятия данной посылки система автоматически добавляет тестовую игру с четырьмя участниками (формат 4×1 с использованием кодемобилей багги): непосредственно тестовой посылкой, стратегией из раздела «Быстрый старт» и двумя стратегиями, которые ничего не делают. Тестовая игра видна только участнику, сделавшему данную тестовую посылку. Базовая длительность такой тестовой игры составляет 2000 тиков. На частоту тестовых посылок действует то же ограничение, что и на частоту обычных посылок. Тестовые игры на частоту создания игр пользователем не влияют.

У игроков есть возможность в специальном визуализаторе просматривать прошедшие игры. Для этого нужно нажать кнопку «Смотреть» в списке игр либо нажать кнопку «Посмотреть игру» на странице игры.

Если вы смотрите игру с участием вашей стратегии и заметили некоторую странность в её поведении, или ваша стратегия делает не то, что вы от неё ожидали, то вы можете воспользоваться специальной утилитой Repeater для воспроизведения локального повтора данной игры. Локальный повтор игры — это возможность запустить стратегию на вашем компьютере так, чтобы она видела игровой мир вокруг себя таким, каким он был при тестировании на сервере. Это поможет вам выполнять отладку, добавлять логирование и наблюдать за реакцией вашей стратегии в каждый момент игры. Для этого скачайте Repeater с сайта CodeRacing 2015 (раздел «Документация» → «Утилита Repeater») и разархивируйте. Для запуска Repeater вам необходимо установленное ПО Java 7+ Runtime Environment. Обратите внимание, что любое взаимодействие вашей стратегии с игровым миром при локальном повторе полностью игнорируется. Это означает, что в каждый момент времени окружающий мир для стратегии в точности совпадает с миром, каким он был в игре при тестировании на сервере и не зависит от того, какие действия ваша стратегия предпринимает. Подробнее об утилите Repeater читайте в соответствующем разделе на сайте.

Помимо всего выше перечисленного у игроков есть возможность запускать простые тестовые игры локально на своём компьютере. Для этого необходимо загрузить архив с утилитой Local runner из раздела сайта «Документация» → «Local runner». Использование данной утилиты позволит вам тестировать свою стратегию в условиях, аналогичных условиям тестовой игры на сайте, но без каких либо ограничений по количеству создаваемых игр. Рендерер для локальных игр заметно отличается от рендерера на сайте. Все игровые объекты в нём отображаются схематично (без использования красочных моделей). Создать локальную тестовую игру очень просто: запустите Local runner с помощью соответствующего скрипта запуска (для Windows или *n*x систем), затем запустите свою стратегию из среды разработки (или любым другим удобным вам способом) и смотрите игру. Во время локальных игр вы можете выполнять отладку своей стратегии, ставить точки останова. Однако следует помнить, что Local runner ожидает отклика от стратегии не более 20 минут. По прошествии этого времени он посчитает стратегию «упавшей» и продолжит работу без неё.

3.2 Управление кодемобилем

Для каждого кодемобиля в вашей команде в начале игры создаётся отдельный экземпляр класса `MyStrategy`, в полях которого стратегия может хранить информацию о данном кодемобиле. Общую для всей команды информацию, в зависимости от языка программирования, можно хранить в статических полях или глобальных переменных.

Управление кодемобилем осуществляется с помощью метода `move` стратегии, который вызывается один раз за тик для каждого кодемобиля. Методу передаются следующие параметры:

- кодемобиль `self`, для которого вызывается метод;
- текущее состояние мира `world`;

- набор игровых констант `game`;
- объект `move`, устанавливая свойства которого, стратегия и определяет поведение кодемобиля.

Реализация клиента-оболочки стратегии на разных языках может отличаться, однако в общем случае не гарантируется, что при разных вызовах метода `move` в качестве параметров ему будут переданы ссылки на одни и те же объекты. Таким образом, нельзя, например, сохранить ссылки на объекты `world` или `player` и получать в следующие тики обновлённую информацию об этих объектах, считывая их поля.

3.3 Примеры реализации

Далее для всех языков программирования приведены простейшие примеры стратегий, которые придают кодемобилю ускорение вперёд, а также используют некоторые расходники. Полную документацию классов и методов для языка Java можно найти в следующих главах.

3.3.1 Пример для Java

```
import model.*;

public final class MyStrategy implements Strategy {
    @Override
    public void move(Car self, World world, Game game, Move move) {
        move.setEnginePower(1.0D);
        move.setThrowProjectile(true);
        move.setSpillOil(true);

        if (world.getTick() > game.getInitialFreezeDurationTicks()) {
            move.setUseNitro(true);
        }
    }
}
```

3.3.2 Пример для C#

```
using System;
using Com.CodeGame.CodeRacing2015.DevKit.CSharpCgdk.Model;

namespace Com.CodeGame.CodeRacing2015.DevKit.CSharpCgdk {
    public sealed class MyStrategy : IStrategy {
        public void Move(Car self, World world, Game game, Move move) {
            move.EnginePower = 1.0D;
            move.IsThrowProjectile = true;
            move.IsSpillOil = true;

            if (world.Tick > game.InitialFreezeDurationTicks) {
                move.IsUseNitro = true;
            }
        }
    }
}
```

3.3.3 Пример для C++

```
#include "MyStrategy.h"

#define PI 3.14159265358979323846
#define _USE_MATH_DEFINES

#include <cmath>
#include <cstdlib>

using namespace model;
using namespace std;

void MyStrategy::move(const Car& self, const World& world, const Game& game, Move& move) {
    move.setEnginePower(1.0);
    move.setThrowProjectile(true);
    move.setSpillOil(true);

    if (world.getTick() > game.getInitialFreezeDurationTicks()) {
        move.setUseNitro(true);
    }
}

MyStrategy::MyStrategy() { }
```

3.3.4 Пример для Python 2

В языке Python 2 имя переменной текущего кодемобиля изменено с «`self`» на «`me`».

```
from model.Car import Car
from model.Game import Game
from model.Move import Move
from model.World import World


class MyStrategy:
    def move(self, me, world, game, move):
        """
        @type me: Car
        @type world: World
        @type game: Game
        @type move: Move
        """
        move.engine_power = 1.0
        move.throw_projectile = True
        move.spill_oil = True

        if world.tick > game.initial_freeze_duration_ticks:
            move.use_nitro = True
```

3.3.5 Пример для Python 3

В языке Python 3 имя переменной текущего кодемобиля изменено с «`self`» на «`me`».

```
from model.Car import Car
from model.Game import Game
from model.Move import Move
from model.World import World


class MyStrategy:
    def move(self, me: Car, world: World, game: Game, move: Move):
        move.engine_power = 1.0
        move.throw_projectile = True
        move.spill_oil = True

        if world.tick > game.initial_freeze_duration_ticks:
            move.use_nitro = True
```

3.3.6 Пример для Pascal

В языке Pascal имя переменной текущего кодемобиля изменено с «`self`» на «`me`».

```
unit MyStrategy;

interface

uses
    StrategyControl, BonusControl, BonusTypeControl, CarControl, CarTypeControl, DirectionControl,
    GameControl, MoveControl, OilSlickControl, PlayerControl, ProjectileControl,
    ProjectileTypeControl, TileTypeControl, TypeControl, WorldControl;

type
    TMyStrategy = class (TStrategy)
    public
        procedure Move(me: TCar; world: TWorld; game: TGame; move: TMove); override;

    end;

implementation

uses
    Math;

procedure TMyStrategy.Move(me: TCar; world: TWorld; game: TGame; move: TMove);
begin
    move.SetEnginePower(1.0);
    move.SetThrowProjectile(true);
    move.SetSpillOil(true);

    if (world.GetTick() > game.GetInitialFreezeDurationTicks()) then begin
        move.SetUseNitro(true);
    end;
end;

end.
```

3.3.7 Пример для Ruby

В языке Ruby имя переменной текущего кодемобиля изменено с «`self`» на «`me`».

```
require './model/car'
require './model/game'
require './model/move'
require './model/world'

class MyStrategy
  # @param [Car] me
  # @param [World] world
  # @param [Game] game
  # @param [Move] move
  def move(me, world, game, move)
    move.engine_power = 1.0
    move.throw_projectile = true
    move.spill_oil = true

    if world.tick > game.initial_freeze_duration_ticks
        move.use_nitro = true
    end
  end
end
```

Глава 4

Package model

Package Contents *Page*

4.1 Classes

4.1.1 CLASS **Bonus**

Класс, определяющий бонус — неподвижный полезный объект. Содержит также все свойства прямоугольного юнита.

DECLARATION

public class Bonus **extends** RectangularUnit

METHODS

- *getType*
  `public BonusType` **`getType`**`( )`
  - **Returns** - Возвращает тип бонуса.

4.1.2 CLASS **BonusType**

Тип бонуса.

DECLARATION

public final class BonusType **extends** Enum

FIELDS

- public static final BonusType REPAIR_KIT
  - Ремкомплект. Полностью устраняет все повреждения кодемобиля при подборе.
- public static final BonusType AMMO_CRATE
  - Ящик с метательными снарядами. При подборе пополняет запас снарядов кодемобиля на единицу. Тип снарядов соответствует типу кодемобиля.

- public static final BonusType NITRO_BOOST
  - Топливо для системы закиси азота. При использовании мгновенно устанавливает мощность двигателя кодемобиля равной значению `game.nitroEnginePowerFactor` и не даёт изменять её в течение `game.useNitroCooldownTicks` тиков.
- public static final BonusType OIL_CANISTER
  - Канистра с мазутом. При использовании на `game.oilSlickLifetime` тиков создаёт позади кодемобиля скользкую круглую область. Радиус области равен `game.oilSlickRadius`, центр находится на продольной оси кодемобиля, а расстояние между ближайшими точками области и кодемобиля составляет `game.oilSlickInitialRange`.
- public static final BonusType PURE_SCORE
  - Баллы в чистом виде. При подборе данного бонуса игроку мгновенно начисляется `game.pureScoreAmount` баллов.

METHODS

- *valueOf*
  `public static BonusType valueOf( String  name )`
- *values*
  `public static BonusType[] values( )`

4.1.3 CLASS Car

Класс, определяющий кодемобиль. Содержит также все свойства прямоугольного юнита.

DECLARATION

---

public class Car

extends RectangularUnit

METHODS

- *getDurability*
  `public double getDurability( )`
  - **Returns** - Возвращает текущую прочность кодемобиля в интервале `[0.0, 1.0]`.
- *getEnginePower*
  `public double getEnginePower( )`
  - **Returns** - Возвращает относительную мощность двигателя кодемобиля. Значение находится в интервале `[-1.0, 1.0]` кроме случаев, когда кодемобиль использует ускорение «нитро».

- *getNextWaypointIndex*
`public int` **getNextWaypointIndex**`( )`
  - **Returns** - Возвращает индекс следующего ключевого тайла в массиве `world.waypoints`.

- *getNextWaypointX*
`public int` **getNextWaypointX**`( )`
  - **Returns** - Возвращает компоненту X позиции следующего ключевого тайла. Конвертировать позицию в точные координаты можно, используя значение `game.trackTileSize`.

- *getNextWaypointY*
`public int` **getNextWaypointY**`( )`
  - **Returns** - Возвращает компоненту Y позиции следующего ключевого тайла. Конвертировать позицию в точные координаты можно, используя значение `game.trackTileSize`.

- *getNitroChargeCount*
`public int` **getNitroChargeCount**`( )`
  - **Returns** - Возвращает количество зарядов для системы закиси азота.

- *getOilCanisterCount*
`public int` **getOilCanisterCount**`( )`
  - **Returns** - Возвращает количество канистр с мазутом.

- *getPlayerId*
`public long` **getPlayerId**`( )`
  - **Returns** - Возвращает идентификатор игрока, которому принадлежит кодемобиль.

- *getProjectileCount*
`public int` **getProjectileCount**`( )`
  - **Returns** - Возвращает количество метательных снарядов.

- *getRemainingNitroCooldownTicks*
`public int` **getRemainingNitroCooldownTicks**`( )`
  - **Returns** - Возвращает количество тиков, по прошествии которого кодемобиль может использовать очередной заряд системы закиси азота, или 0, если кодемобиль может совершить данное действие в текущий тик.

- *getRemainingNitroTicks*
`public int` **getRemainingNitroTicks**`( )`
  - **Returns** - Возвращает количество оставшихся тиков действия системы закиси азота.

- *getRemainingOilCooldownTicks*
`public int` **getRemainingOilCooldownTicks**`( )`
  - **Returns** - Возвращает количество тиков, по прошествии которого кодемобиль может разлить очередную лужу мазута, или 0, если кодемобиль может совершить данное действие в текущий тик.

- *getRemainingOiledTicks*
`public int` **getRemainingOiledTicks**`( )`
  - **Returns** - Возвращает количество тиков, оставшихся до полного высыхания кодемобиля, попавшего в лужу мазута.

- *getRemainingProjectileCooldownTicks*
`public int` **getRemainingProjectileCooldownTicks**`( )`

- **Returns** - Возвращает количество тиков, по прошествии которого кодемобиль может запустить очередной снаряд, или `0`, если кодемобиль может совершить данное действие в текущий тик.

- *getTeammateIndex*
`public int` **getTeammateIndex**`( )`

  - **Returns** - Возвращает 0-индексированный номер кодемобиля среди юнитов одного игрока.

- *getType*
`public CarType` **getType**`( )`

  - **Returns** - Возвращает тип кодемобиля.

- *getWheelTurn*
`public double` **getWheelTurn**`( )`

  - **Returns** - Возвращает относительный угол поворота колёс (или руля, что эквивалентно) кодемобиля в интервале `[-1.0, 1.0]`.

- *isFinishedTrack*
`public boolean` **isFinishedTrack**`( )`

  - **Returns** - Возвращает `true`, если и только если данный кодемобиль финишировал. Финишировавший кодемобиль перестаёт управляться игроком, а также участвовать в столкновениях с другими юнитами.

- *isTeammate*
`public boolean` **isTeammate**`( )`

  - **Returns** - Возвращает `true`, если и только если данный кодемобиль принадлежит вам.

4.1.4 CLASS **CarType**

Тип кодемобиля.

В Раунде 1 чемпионата стратегия игрока управляет одним кодемобилем типа `BUGGY`. В гонке участвуют `4` игрока.

В Раунде 2 чемпионата стратегия игрока управляет одним кодемобилем типа `JEEP`. В гонке участвуют `4` игрока.

В Финале в распоряжении стратегии игрока находится по одному кодемобилю каждого типа. В гонке участвуют `2` игрока.

DECLARATION

---

public final class CarType

extends Enum

FIELDS

- public static final CarType BUGGY
  - Багги. Стреляет тремя небольшими шайбами, расходящимися под небольшим углом. Немного легче джипа.
- public static final CarType JEEP
  - Джип. Стреляет большими массивными шинами, отскакивающими от машин и границ трассы. Немного тяжелее багги.

METHODS

- *valueOf*
  `public static CarType valueOf( String  name )`
- *values*
  `public static CarType[] values( )`

4.1.5 CLASS **CircularUnit**

Базовый класс для определения круглых объектов. Содержит также все свойства юнита.

DECLARATION

public abstract class CircularUnit **extends** Unit

METHODS

- *getRadius*
  `public double getRadius( )`
  - **Returns** - Возвращает радиус объекта.

4.1.6 CLASS **Direction**

Направление.

DECLARATION

public final class Direction

extends Enum

FIELDS

- public static final Direction LEFT
  - Налево/слева.
- public static final Direction RIGHT
  - Направо/справа.
- public static final Direction UP
  - Вверх/сверху.
- public static final Direction DOWN
  - Вниз/снизу.

METHODS

- *valueOf*
  `public static Direction valueOf( String  name )`
- *values*
  `public static Direction[] values( )`

4.1.7 CLASS **Game**

Предоставляет доступ к различным игровым константам.

DECLARATION

public class Game

extends Object

- *getBonusMass*
`public double` **getBonusMass**`( )`
  - **Returns** - Возвращает массу бонуса.

- *getBonusSize*
`public double` **getBonusSize**`( )`
  - **Returns** - Возвращает размер (ширину и высоту) бонуса.

- *getBuggyEngineForwardPower*
`public double` **getBuggyEngineForwardPower**`( )`
  - **Returns** - Возвращает максимальную мощность двигателя кодемобиля типа багги (`CarType.BUGGY`) в направлении, совпадающем с направлением кодемобиля.

- *getBuggyEngineRearPower*
`public double` **getBuggyEngineRearPower**`( )`
  - **Returns** - Возвращает максимальную мощность двигателя кодемобиля типа багги (`CarType.BUGGY`) в направлении, противоположном направлению кодемобиля.

- *getBuggyMass*
`public double` **getBuggyMass**`( )`
  - **Returns** - Возвращает массу кодемобиля типа багги (`CarType.BUGGY`).

- *getBurningTimeDurationFactor*
`public double` **getBurningTimeDurationFactor**`( )`
  - **Returns** - Возвращает коэффициент, определяющий количество тиков до завершения игры после финиширования трассы очередным кодемобилем. Для получения более подробной информации смотрите документацию к `world.lastTickIndex`.

- *getCarAngularSpeedFactor*
`public double` **getCarAngularSpeedFactor**`( )`
  - **Returns** - Возвращает коэффициент, используемый для вычисления составляющей угловой скорости кодемобиля, порождаемой движением кодемобиля при ненулевом относительном угле поворота колёс. Для получения более подробной информации смотрите документацию к `move.wheelTurn`.

- *getCarCrosswiseMovementFrictionFactor*
`public double` **getCarCrosswiseMovementFrictionFactor**`( )`
  - **Returns** - Возвращает абсолютную потерю составляющей скорости кодемобиля, направленной вдоль поперечной оси кодемобиля, за один тик.

- *getCarDamageScoreFactor*
`public double` **getCarDamageScoreFactor**`( )`
  - **Returns** - Возвращает количество баллов, зарабатываемых кодемобилем при нанесении `1.0` урона кодемобилю другого игрока. При нанесении меньшего урона количество баллов пропорционально падает. Результат всегда округляется в меньшую сторону.

- *getCarEliminationScore*
`public int` **getCarEliminationScore**`( )`

- **Returns** - Возвращает количество баллов, зарабатываемых кодемобилем при уничтожении кодемобиля другого игрока.

- *getCarEnginePowerChangePerTick*
`public double` **getCarEnginePowerChangePerTick**`( )`

  - **Returns** - Возвращает максимальное значение, на которое может измениться относительная мощность двигателя кодемобиля (`car.enginePower`) за один тик.

- *getCarHeight*
`public double` **getCarHeight**`( )`

  - **Returns** - Возвращает высоту кодемобиля.

- *getCarLengthwiseMovementFrictionFactor*
`public double` **getCarLengthwiseMovementFrictionFactor**`( )`

  - **Returns** - Возвращает абсолютную потерю составляющей скорости кодемобиля, направленной вдоль продольной оси кодемобиля, за один тик.

- *getCarMovementAirFrictionFactor*
`public double` **getCarMovementAirFrictionFactor**`( )`

  - **Returns** - Возвращает относительную потерю модуля скорости кодемобиля за один тик.

- *getCarReactivationTimeTicks*
`public int` **getCarReactivationTimeTicks**`( )`

  - **Returns** - Возвращает длительность интервала в тиках, по прошествии которого сильно повреждённый кодемобиль (значение `car.durability` равно нулю) будет восстановлен.

- *getCarRotationAirFrictionFactor*
`public double` **getCarRotationAirFrictionFactor**`( )`

  - **Returns** - Возвращает относительную потерю модуля угловой скорости кодемобиля за один тик. Относительная потеря применяется только к составляющей угловой скорости кодемобиля, не порождаемой движением кодемобиля при ненулевом относительном угле поворота колёс. Для получения более подробной информации смотрите документацию к `move.wheelTurn`.

- *getCarRotationFrictionFactor*
`public double` **getCarRotationFrictionFactor**`( )`

  - **Returns** - Возвращает абсолютную потерю модуля угловой скорости кодемобиля за один тик. Абсолютная потеря применяется только к составляющей угловой скорости кодемобиля, не порождаемой движением кодемобиля при ненулевом относительном угле поворота колёс. Для получения более подробной информации смотрите документацию к `move.wheelTurn`.

- *getCarWheelTurnChangePerTick*
`public double` **getCarWheelTurnChangePerTick**`( )`

  - **Returns** - Возвращает максимальное значение, на которое может измениться относительный угол поворота колёс кодемобиля (`car.wheelTurn`) за один тик.

- *getCarWidth*
`public double` **getCarWidth**`( )`

  - **Returns** - Возвращает ширину кодемобиля.

- *getFinishLapScore*
`public int` **getFinishLapScore**`( )`

- **Returns** - Возвращает количество баллов, зарабатываемых кодемобилем при прохождении одного круга. Баллы начисляются не единовременно, а постепенно, по мере прохождения ключевых точек.

- *getFinishTrackScores*
`public int[] getFinishTrackScores( )`

  - **Returns** - Возвращает 0-индексированный массив, содержащий количество баллов, зарабатываемых кодемобилями при завершении трассы. Кодемобиль, финишировавший первым, приносит владельцу `finishTrackScores[0]` баллов, вторым — `finishTrackScores[1]` и так далее.

- *getInitialFreezeDurationTicks*
`public int getInitialFreezeDurationTicks( )`

  - **Returns** - Возвращает количество тиков в начале игры, в течение которых кодемобиль не может изменять своё положение. Значение является составной частью выражения для нахождения базовой длительности игры (`game.tickCount`).

- *getJeepEngineForwardPower*
`public double getJeepEngineForwardPower( )`

  - **Returns** - Возвращает максимальную мощность двигателя кодемобиля типа джип (`CarType.JEEP`) в направлении, совпадающем с направлением кодемобиля.

- *getJeepEngineRearPower*
`public double getJeepEngineRearPower( )`

  - **Returns** - Возвращает максимальную мощность двигателя кодемобиля типа джип (`CarType.JEEP`) в направлении, противоположном направлению кодемобиля.

- *getJeepMass*
`public double getJeepMass( )`

  - **Returns** - Возвращает массу кодемобиля типа джип (`CarType.JEEP`).

- *getLapCount*
`public int getLapCount( )`

  - **Returns** - Возвращает количество кругов (циклов прохождения списка ключевых точек `world.waypoints`), которое необходимо пройти для завершения трассы.

- *getLapTickCount*
`public int getLapTickCount( )`

  - **Returns** - Возвращает количество тиков, которое выделяется кодемобилям на прохождение одного круга. Значение является составной частью выражения для нахождения базовой длительности игры (`game.tickCount`) и не используется в целях ограничения на прохождение одного отдельного круга.

- *getLapWaypointsSummaryScoreFactor*
`public double getLapWaypointsSummaryScoreFactor( )`

  - **Returns** - Возвращает долю от баллов за круг (`game.finishLapScore`), которую кодемобиль заработает при прохождении всех ключевых точек круга, кроме последней. Баллы равномерно распределены по ключевым точкам.

- *getMaxOiledStateDurationTicks*
`public int getMaxOiledStateDurationTicks( )`

- **Returns** - Возвращает максимально возможную длительность высыхания кодемобиля, центр которого попал в лужу мазута. При этом, длительность высыхания лужа мазута сокращается на то же количество тиков. Таким образом, реальная длительность высыхания кодемобиля не может превышать оставшуюся длительность высыхания лужи.

- *getNitroDurationTicks*
`public int` **getNitroDurationTicks**`( )`

  - **Returns** - Возвращает длительность ускорения «нитро»в тиках.

- *getNitroEnginePowerFactor*
`public double` **getNitroEnginePowerFactor**`( )`

  - **Returns** - Возвращает относительную мощность двигателя кодемобиля, мгновенно устанавливаемую при использовании системы закиси азота для ускорения кодемобиля.

- *getOilSlickInitialRange*
`public double` **getOilSlickInitialRange**`( )`

  - **Returns** - Возвращает расстояние между ближайшими точками лужи мазута и кодемобиля при использовании канистры с мазутом.

- *getOilSlickLifetime*
`public int` **getOilSlickLifetime**`( )`

  - **Returns** - Возвращает длительность высыхания лужи мазута в тиках.

- *getOilSlickRadius*
`public double` **getOilSlickRadius**`( )`

  - **Returns** - Возвращает радиус лужи мазута.

- *getPureScoreAmount*
`public int` **getPureScoreAmount**`( )`

  - **Returns** - Возвращает количество баллов, мгновенно получаемых игроком, кодемобиль которого подобрал бонусные баллы (`BonusType.PURE_SCORE`).

- *getRandomSeed*
`public long` **getRandomSeed**`( )`

  - **Returns** - Возвращает некоторое число, которое ваша стратегия может использовать для инициализации генератора случайных чисел. Данное значение имеет рекомендательный характер, однако позволит более точно воспроизводить прошедшие игры.

- *getSideWasherAngle*
`public double` **getSideWasherAngle**`( )`

  - **Returns** - Возвращает модуль отклонения направления полёта двух шайб от направления кодемобиля. Направление третьей шайбы совпадает с направлением кодемобиля.

- *getSpillOilCooldownTicks*
`public int` **getSpillOilCooldownTicks**`( )`

  - **Returns** - Возвращает длительность задержки в тиках, применяемой к кодемобилю после использования им канистры с мазутом. В течение этого времени кодемобиль не может разлить ещё одну канистру.

- *getThrowProjectileCooldownTicks*
`public int` **getThrowProjectileCooldownTicks**`( )`

  - **Returns** - Возвращает длительность задержки в тиках, применяемой к кодемобилю после метания им снаряда. В течение этого времени кодемобиль не может метать новые снаряды.

- *getTickCount*
`public int` **getTickCount**`( )`

  - **Returns** - Возвращает базовую длительность игры в тиках. Реальная длительность может отличаться от этого значения в меньшую сторону. Поле может быть определено как `game.initialFreezeDurationTicks + game.lapCount * game.lapTickCount`. Значение поля не меняется в процессе игры. Эквивалентно `world.tickCount`.

- *getTireDamageFactor*
`public double` **getTireDamageFactor**`( )`

  - **Returns** - Возвращает количество урона, которое шина нанесёт неподвижно стоящему кодемобилю при попадании в него с начальной скоростью (`game.tireInitialSpeed`) и под прямым углом к поверхности кодемобиля. Движение кодемобиля в направлении, совпадающем с направлением движения шины, уменьшает урон, движение в противоположном направлении — увеличивает.

- *getTireDisappearSpeedFactor*
`public double` **getTireDisappearSpeedFactor**`( )`

  - **Returns** - Возвращает отношение текущей скорости шины к начальной (`game.tireInitialSpeed`), при превышении которого в момент столкновения с другим объектом шина отскакивает и продолжает свой полёт. В противном случае шина убирается из игрового мира.

- *getTireInitialSpeed*
`public double` **getTireInitialSpeed**`( )`

  - **Returns** - Возвращает начальную скорость шины (`ProjectileType.TIRE`).

- *getTireMass*
`public double` **getTireMass**`( )`

  - **Returns** - Возвращает массу шины (`ProjectileType.TIRE`).

- *getTireRadius*
`public double` **getTireRadius**`( )`

  - **Returns** - Возвращает радиус шины (`ProjectileType.TIRE`).

- *getTrackTileMargin*
`public double` **getTrackTileMargin**`( )`

  - **Returns** - Возвращает отступ от границы тайла до границы прямого участка трассы, проходящего через этот тайл. Радиусы всех закруглённых сочленений участков трассы также равны этому значению.

- *getTrackTileSize*
`public double` **getTrackTileSize**`( )`

  - **Returns** - Возвращает размер (ширину и высоту) одного тайла.

- *getUseNitroCooldownTicks*
`public int` **getUseNitroCooldownTicks**`( )`

  - **Returns** - Возвращает длительность задержки в тиках, применяемой к кодемобилю после использования им ускорения «нитро». В течение этого времени кодемобиль не может повторно использовать систему закиси азота.

- *getWasherDamage*
`public double` **getWasherDamage**`( )`

- **Returns** - Возвращает урон шайбы (`ProjectileType.WASHER`).

- *getWasherInitialSpeed*
`public double` **getWasherInitialSpeed**`( )`

  - **Returns** - Возвращает начальную скорость шайбы (`ProjectileType.WASHER`).

- *getWasherMass*
`public double` **getWasherMass**`( )`

  - **Returns** - Возвращает массу шайбы (`ProjectileType.WASHER`).

- *getWasherRadius*
`public double` **getWasherRadius**`( )`

  - **Returns** - Возвращает радиус шайбы (`ProjectileType.WASHER`).

- *getWorldHeight*
`public int` **getWorldHeight**`( )`

  - **Returns** - Возвращает высоту игрового мира в тайлах.

- *getWorldWidth*
`public int` **getWorldWidth**`( )`

  - **Returns** - Возвращает ширину игрового мира в тайлах.

4.1.8 CLASS **Move**

Стратегия игрока может управлять кодемобилем посредством установки свойств объекта данного класса.

DECLARATION

public class Move **extends** Object

METHODS

- *getEnginePower*
`public double` **getEnginePower**`( )`

  - **Returns** - Возвращает текущую установку режима работы двигателя кодемобиля.

- *getWheelTurn*
`public double` **getWheelTurn**`( )`

  - **Returns** - Возвращает текущий относительный угол поворота колёс кодемобиля.

- *isBrake*
`public boolean` **isBrake**`( )`

- **Returns** - Возвращает текущее положение педали тормоза.

- *isSpillOil*
`public boolean` **isSpillOil**`( )`

  - **Returns** - Возвращает текущее значение указания разлить канистру с мазутом.

- *isThrowProjectile*
`public boolean` **isThrowProjectile**`( )`

  - **Returns** - Возвращает текущее значение указания метнуть снаряд.

- *isUseNitro*
`public boolean` **isUseNitro**`( )`

  - **Returns** - Возвращает текущее значение указания использовать «нитро».

- *setBrake*
`public void` **setBrake**`( boolean` **brake** `)`

  - **Usage**
    * Задаёт текущее положение педали тормоза.
      При утопленной педали тормоза значение силы трения вдоль направления, совпадающего с углом поворота кодемобиля, возрастает с `game.carLengthwiseMovementFrictionFactor` до `game.carCrosswiseMovementFrictionFactor`.

- *setEnginePower*
`public void` **setEnginePower**`( double` **enginePower** `)`

  - **Usage**
    * Задаёт установку режима работы двигателя кодемобиля.
      Установка режима работы является относительной и должна лежать в интервале от `-1.0` до `1.0`. Значения, выходящие за указанный интервал, будут приведены к ближайшей его границе.
      Реальный режим работы двигателя может отличаться от установки, так как его изменение происходит не мгновенно, а со скоростью не более `game.carEnginePowerChangePerTick` за тик. Режим работы двигателя фактически определяет ускорение в направлении, совпадающем с углом поворота кодемобиля. Абсолютное значение ускорения равномерно изменяется на интервале от `-1.0` до `0.0` и на интервале от `0.0` до `1.0`.

- *setSpillOil*
`public void` **setSpillOil**`( boolean` **spillOil** `)`

  - **Usage**
    * Устанавливает значение указания разлить канистру с мазутом.
      Указание может быть проигнорировано, если у кодемобиля не осталось канистр с мазутом либо прошло менее `game.spillOilCooldownTicks` тиков с момента предыдущего использования данного действия.

- *setThrowProjectile*
`public void` **setThrowProjectile**`( boolean` **throwProjectile** `)`

  - **Usage**
    * Устанавливает значение указания метнуть снаряд.
      Указание может быть проигнорировано, если у кодемобиля не осталось снарядов либо прошло менее `game.throwProjectileCooldownTicks` тиков с момента запуска предыдущего снаряда.

- *setUseNitro*
`public void` **setUseNitro**`( boolean` **useNitro** `)`

- **Usage**
  - Устанавливает значение указания использовать «нитро».
    Указание может быть проигнорировано, если у кодемобиля не осталось зарядов для системы закиси азота либо прошло менее `game.useNitroCooldownTicks` тиков с момента предыдущего ускорения.

- *setWheelTurn*
  `public void` **setWheelTurn**`( double` **wheelTurn** `)`
  - **Usage**
    - Задаёт относительный угол поворота колёс (или руля, что эквивалентно) кодемобиля.
      Относительный угол должен лежать в интервале от `-1.0` до `1.0`. Значения, выходящие за указанный интервал, будут приведены к ближайшей его границе.
      Реальный поворот колёс может отличаться от установки, так как его изменение происходит не мгновенно, а со скоростью не более `game.carWheelTurnChangePerTick` за тик. Поворот колёс создаёт добавочную угловую скорость кодемобиля (помимо угловой скорости, вызванной соударениями объектов и другими причинами), значение которой прямо пропорционально текущему относительному углу поворота колёс кодемобиля (`car.wheelTurn`), коэффициенту `game.carAngularSpeedFactor`, а также скалярному произведению вектора скорости кодемобиля и единичного вектора, направление которого совпадает с направлением кодемобиля.

4.1.9 CLASS **OilSlick**

Класс, определяющий лужу мазута. Содержит также все свойства круглого юнита.

DECLARATION

---

public class OilSlick **extends** CircularUnit

METHODS

- *getRemainingLifetime*
  `public int` **getRemainingLifetime**`( )`
  - **Returns** - Возвращает количество тиков, по прошествии которого лужа мазута полностью высохнет.

4.1.10 CLASS **Player**

Содержит данные о текущем состоянии игрока.

DECLARATION

public class Player

extends Object

METHODS

- *getId*
  `public long` **getId**`( )`
  - **Returns** - Возвращает уникальный идентификатор игрока.

- *getName*
  `public String` **getName**`( )`
  - **Returns** - Возвращает имя игрока.

- *getScore*
  `public int` **getScore**`( )`
  - **Returns** - Возвращает количество баллов, набранное игроком.

- *isMe*
  `public boolean` **isMe**`( )`
  - **Returns** - Возвращает `true` в том и только в том случае, если этот игрок ваш.

- *isStrategyCrashed*
  `public boolean` **isStrategyCrashed**`( )`
  - **Returns** - Возвращает специальный флаг — показатель того, что стратегия игрока «упала». Более подробную информацию можно найти в документации к игре.

4.1.11 CLASS **Projectile**

Класс, определяющий метательный снаряд. Содержит также все свойства круглого юнита.

DECLARATION

public class Projectile

extends CircularUnit

METHODS

- *getCarId*
  `public long` **getCarId**`( )`
  - **Returns** - Возвращает идентификатор кодемобиля, выпустившего данный снаряд.
- *getPlayerId*
  `public long` **getPlayerId**`( )`
  - **Returns** - Возвращает идентификатор игрока, кодемобиль которого выпустил данный снаряд.
- *getType*
  `public ProjectileType` **getType**`( )`
  - **Returns** - Возвращает тип метательного снаряда.

4.1.12 CLASS **ProjectileType**

Тип метательного снаряда.

DECLARATION

public final class ProjectileType

extends Enum

FIELDS

- public static final ProjectileType WASHER
  - Небольшой и лёгкий метательный снаряд для кодемобилей типа багги. Свободно пролетает сквозь ограждение трассы. Наносит относительно небольшой урон при попадании в кодемобиль и сразу после этого исчезает.
- public static final ProjectileType TIRE
  - Метательный снаряд, сопоставимый по размеру и массе с кодемобилем. Используется кодемобилями типа джип. Отражается как от кодемобилей, так и от границ трассы. Если скорость снаряда при столкновении меньше значения `game.tireDisappearSpeedFactor * game.tireInitialSpeed`, то сразу после столкновения он исчезает.

METHODS

- *valueOf*
  `public static ProjectileType` **valueOf**`( String` **name** `)`
- *values*
  `public static ProjectileType[]` **values**`( )`

4.1.13 CLASS **RectangularUnit**

Базовый класс для определения прямоугольных объектов. Содержит также все свойства юнита.

DECLARATION

public abstract class RectangularUnit **extends** Unit

METHODS

- *getHeight*
  `public double` **getHeight**`( )`
  - **Returns** - Возвращает высоту объекта.
- *getWidth*
  `public double` **getWidth**`( )`
  - **Returns** - Возвращает ширину объекта.

4.1.14 CLASS **TileType**

Тип тайла.

DECLARATION

public final class TileType **extends** Enum

- public static final TileType EMPTY
  - Пустой тайл.
- public static final TileType VERTICAL
  - Тайл с прямым вертикальным участком дороги.
- public static final TileType HORIZONTAL
  - Тайл с прямым горизонтальным участком дороги.
- public static final TileType LEFT_TOP_CORNER
  - Тайл, выполняющий роль сочленения двух других тайлов: справа и снизу от данного тайла.
- public static final TileType RIGHT_TOP_CORNER
  - Тайл, выполняющий роль сочленения двух других тайлов: слева и снизу от данного тайла.
- public static final TileType LEFT_BOTTOM_CORNER
  - Тайл, выполняющий роль сочленения двух других тайлов: справа и сверху от данного тайла.
- public static final TileType RIGHT_BOTTOM_CORNER
  - Тайл, выполняющий роль сочленения двух других тайлов: слева и сверху от данного тайла.
- public static final TileType LEFT_HEADED_T
  - Тайл, выполняющий роль сочленения трёх других тайлов: слева, снизу и сверху от данного тайла.
- public static final TileType RIGHT_HEADED_T
  - Тайл, выполняющий роль сочленения трёх других тайлов: справа, снизу и сверху от данного тайла.
- public static final TileType TOP_HEADED_T
  - Тайл, выполняющий роль сочленения трёх других тайлов: слева, справа и сверху от данного тайла.
- public static final TileType BOTTOM_HEADED_T
  - Тайл, выполняющий роль сочленения трёх других тайлов: слева, справа и снизу от данного тайла.
- public static final TileType CROSSROADS
  - Тайл, выполняющий роль сочленения четырёх других тайлов: со всех сторон от данного тайла.
- public static final TileType UNKNOWN
  - Тип тайла пока не известен.

- *valueOf*
  `public static TileType valueOf( String  name )`
- *values*
  `public static TileType[] values( )`

4.1.15 CLASS **Unit**

Базовый класс для определения объектов («юнитов») на игровом поле.

DECLARATION

public abstract class Unit

extends Object

METHODS

- *getAngle*
  `public final double` **getAngle**`( )`
  - **Returns** - Возвращает угол поворота объекта в радианах. Нулевой угол соответствует направлению оси абсцисс. Положительные значения соответствуют повороту по часовой стрелке.

- *getAngleTo*
  `public double` **getAngleTo**`( double  x, double  y )`
  - **Parameters**
    - `x` - X-координата точки.
    - `y` - Y-координата точки.
  - **Returns** - Возвращает ориентированный угол `[-PI, PI]` между направлением данного объекта и вектором из центра данного объекта к указанной точке.

- *getAngleTo*
  `public double` **getAngleTo**`( Unit  unit )`
  - **Parameters**
    - `unit` - Объект, к центру которого необходимо определить угол.
  - **Returns** - Возвращает ориентированный угол `[-PI, PI]` между направлением данного объекта и вектором из центра данного объекта к центру указанного объекта.

- *getAngularSpeed*
  `public double` **getAngularSpeed**`( )`
  - **Returns** - Возвращает скорость вращения объекта. Положительные значения соответствуют вращению по часовой стрелке.

- *getDistanceTo*
  `public double` **getDistanceTo**`( double  x, double  y )`
  - **Parameters**
    - `x` - X-координата точки.
    - `y` - Y-координата точки.
  - **Returns** - Возвращает расстояние до точки от центра данного объекта.

- *getDistanceTo*
  `public double` **getDistanceTo**`( Unit  unit )`

- **Parameters**
  - `unit` - Объект, до центра которого необходимо определить расстояние.
- **Returns** - Возвращает расстояние от центра данного объекта до центра указанного объекта.

- *getId*
`public long` **getId**`( )`
  - **Returns** - Возвращает уникальный идентификатор объекта.

- *getMass*
`public double` **getMass**`( )`
  - **Returns** - Возвращает массу объекта в единицах массы.

- *getSpeedX*
`public final double` **getSpeedX**`( )`
  - **Returns** - Возвращает X-составляющую скорости объекта. Ось абсцисс направлена слева направо.

- *getSpeedY*
`public final double` **getSpeedY**`( )`
  - **Returns** - Возвращает Y-составляющую скорости объекта. Ось ординат направлена сверху вниз.

- *getX*
`public final double` **getX**`( )`
  - **Returns** - Возвращает X-координату центра объекта. Ось абсцисс направлена слева направо.

- *getY*
`public final double` **getY**`( )`
  - **Returns** - Возвращает Y-координату центра объекта. Ось ординат направлена сверху вниз.

4.1.16 CLASS **World**

Этот класс описывает игровой мир. Содержит также описания всех игроков и игровых объектов («юнитов»).

DECLARATION

---

public class World **extends** Object

- *getBonuses*
`public Bonus[] getBonuses( )`
  - **Returns** - Возвращает список бонусов (в случайном порядке). После каждого тика объекты, задающие бонусы, пересоздаются.

- *getCars*
`public Car[] getCars( )`
  - **Returns** - Возвращает список кодемобилей (в случайном порядке). После каждого тика объекты, задающие кодемобили, пересоздаются.

- *getHeight*
`public int getHeight( )`
  - **Returns** - Возвращает высоту мира в тайлах.

- *getLastTickIndex*
`public int getLastTickIndex( )`
  - **Returns** - Возвращает номер последнего тика игры. Сразу после старта содержит значение `tickCount - 1`. В момент завершения трассы любым из кодемобилей получает значение `min(tickCount - 1, tick + max(floor(game.burningTimeDurationFactor * game.lapTickCount), 1))`. Таким образом, кодемобиль, отставший от идущего впереди более, чем на `game.burningTimeDurationFactor` кругов, рискует не успеть вообще добраться до финиша.
  Игра может закончиться раньше, чем наступит `lastTickIndex`, если для каждого игрока выполняется одно из двух условий: стратегия игрока «упала», либо все его кодемобили финишировали.

- *getMapName*
`public String getMapName( )`
  - **Returns** - Возвращает краткое уникальное название трассы.

- *getMyPlayer*
`public Player getMyPlayer( )`
  - **Returns** - Возвращает вашего игрока.

- *getOilSlicks*
`public OilSlick[] getOilSlicks( )`
  - **Returns** - Возвращает список масляных луж (в случайном порядке). После каждого тика объекты, задающие лужи, пересоздаются.

- *getPlayers*
`public Player[] getPlayers( )`
  - **Returns** - Возвращает список игроков (в случайном порядке). После каждого тика объекты, задающие игроков, пересоздаются.

- *getProjectiles*
`public Projectile[] getProjectiles( )`
  - **Returns** - Возвращает список снарядов (в случайном порядке). После каждого тика объекты, задающие снаряды, пересоздаются.

- *getStartingDirection*
  `public Direction` **getStartingDirection**`( )`
  - **Returns** - Направление кодемобиля в начале игры.

- *getTick*
  `public int` **getTick**`( )`
  - **Returns** - Возвращает номер текущего тика.

- *getTickCount*
  `public int` **getTickCount**`( )`
  - **Returns** - Возвращает базовую длительность игры в тиках. Реальная длительность может отличаться от этого значения в меньшую сторону. Поле может быть определено как `game.initialFreezeDurationTicks + game.lapCount * game.lapTickCount`. Значение поля не меняется в процессе игры. Эквивалентно `game.tickCount`.

- *getTilesXY*
  `public TileType[][]` **getTilesXY**`( )`
  - **Returns** - Возвращает двумерный массив тайлов, где первое измерение — это позиция X, а второе — Y. Конвертировать позицию в точные координаты можно, используя значение `game.trackTileSize`.

- *getWaypoints*
  `public int[][]` **getWaypoints**`( )`
  - **Returns** - Возвращает массив ключевых тайлов. Каждый тайл задаётся массивом длины 2, где элемент с индексом `0` содержит позицию X, а элемент с индексом `1` — позицию Y. Конвертировать позицию в точные координаты можно, используя значение `game.trackTileSize`. Для прохождения круга кодемобилю необходимо посещать тайлы в указанном порядке. Ключевой тайл с индексом `0` является одновременно начальным тайлом трассы и конечным тайлом каждого круга. Считается, что кодемобиль посетил ключевой тайл, если центр кодемобиля пересёк границу этого тайла.

- *getWidth*
  `public int` **getWidth**`( )`
  - **Returns** - Возвращает ширину мира в тайлах.

Глава 5

Package <none>

Package Contents *Page*

Interfaces

Strategy 49

Стратегия — интерфейс, содержащий описание методов искусственного интеллекта кодемобиля.

5.1 Interfaces

5.1.1 INTERFACE Strategy

Стратегия — интерфейс, содержащий описание методов искусственного интеллекта кодемобиля. Каждая пользовательская стратегия должна реализовывать этот интерфейс. Может отсутствовать в некоторых языковых пакетах, если язык не поддерживает интерфейсы.

DECLARATION

```
public interface Strategy
```

METHODS

- *move*
  `public void` **move**`( Car` **self**`, World` **world**`, Game` **game**`, Move` **move** `)`
  - **Usage**
    - Основной метод стратегии, осуществляющий управление кодемобилем. Вызывается каждый тик для каждого кодемобиля.
  - **Parameters**
    - `self` - Кодемобиль, которым данный метод будет осуществлять управление.
    - `world` - Текущее состояние мира.
    - `game` - Различные игровые константы.
    - `move` - Результатом работы метода является изменение полей данного объекта.