CodeWars 2017

Правила

Версия 0.9.0 BETA

Ноябрь — декабрь, 2017

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.

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 7 ноября 2017 года до 24.00 часов 24 декабря 2017 года по Московскому времени.

Первая неделя (с 00.00 часов 7 ноября 2017 года до 24.00 часов 12 ноября 2017 года) и четвёртая неделя (с 00.00 часов 27 ноября 2017 года до 24.00 часов 3 декабря 2017 года) Конкурса являются тестовыми. 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 25 ноября 2017 года до 24 часов 00 минут 26 ноября 2017 года;
- второй этап — с 00 часов 00 минут 9 декабря 2017 года до 24 часов 00 минут 10 декабря 2016 года;
- третий этап (заключительный) — с 00 часов 00 минут 16 декабря 2017 года до 24 часов 00 минут 17 декабря 2017 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 7 ноября 2017 года до 24.00 часов 24 декабря 2017 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 главе 2 данного документа.

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 главе 2 данного документа.

Призовой фонд Конкурса формируется за счет средств Организатора конкурса.

Призовой фонд:

- 1 место — Apple Macbook Pro;
- 2 место — Apple Macbook Air;
- 3 место — Apple iPad;
- 4 место — Samsung Gear S3;
- 5 место — WD My Cloud 6 TB;
- 6 место — WD My Passport Ultra 4TB;
- 1-6 места в Песочнице — WD My Passport Ultra 2TB.

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 CodeWars 2017

2.1 Общие положения игры и правила проведения турнира

Данное соревнование предоставляет вам возможность проверить свои навыки программирования, создав искусственный интеллект (стратегию), управляющий большим количеством боевых единиц (техники) в специальном игровом мире (подробнее об особенностях мира CodeWars 2017 можно узнать в следующих пунктах этой главы). Специфической особенностью задачи этого года является то, что набор действий, доступных вашей стратегии, схож с возможностями управления в обычных компьютерных играх жанра RTS. Также есть ограничение на количество действий в единицу игрового времени. В каждой игре вам будет противостоять стратегия другого игрока. Для победы необходимо набрать больше баллов, чем у вашего оппонента. Баллы начисляются за различные игровые действия. Разумеется, за полное уничтожение противника даётся значительное количество баллов, что почти полностью нивелирует другие достижения в процессе игры. Сохраняется теоретическая возможность уничтожить противника, но в то же время проиграть по баллам, однако на практике такое почти невозможно. Количество баллов, полученных в процессе игры, становится более важным, если ни одному из участников не удалось добиться полной победы за отведённое на игру время.

Сражение происходит на различных типах местности и при различных погодных условиях, влияющих на некоторые параметры техники. В некоторых режимах игры на карте могут присутствовать нейтральные сооружения, захватывая которые стратегия получает возможность производить новую технику или обретает другие игровые преимущества. Игры последнего этапа турнира проводятся в условиях частичной видимости.

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 сильно упадёт после следующей игры в Песочнице, однако по мере дальнейшего 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 места (система, аналогичная используемой в чемпионате «Формула-1»). Если два 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 два участника делят первое место, то каждый из них получит половину от суммы единиц рейтинга за первое и второе места. При делении округление всегда совершается в меньшую сторону. Более подробная информация об этапах турнира будет предоставлена 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 большинства участников он остаётся постоянной величиной. Существует ряд критериев, по которым интервал участия в квалификационных играх может быть увеличен для конкретного игрока. За каждую N-ю полную неделю, прошедшую с момента отправки игроком последней стратегии, интервал участия для этого игрока увеличивается на N базовых интервалов тестирования. Учитываются только принятые системой стратегии. За каждое «падение» стратегии в 10 последних играх в Песочнице начисляется дополнительный штраф, равный 20% от базового интервала тестирования. Подробнее о причинах «падения» стратегии можно узнать в следующих разделах. Интервал участия игрока в Песочнице не может стать больше суток.

Игры в Песочнице проходят по набору правил, соответствующему 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 попадут 1080 участников с наибольшим рейтингом на момент начала этого этапа турнира (при равенстве рейтинга приоритет отдаётся игроку, раньше отправившему последнюю версию своей стратегии), а также дополнительный набор в следующие этапы турнира, включая Финал.

Этапы турнира:

- В **Раунде 1** вам предстоит изучить правила игры и освоить управление большим количеством юнитов. В начале игры вам даётся 500 единиц техники. Вашему оппоненту даётся такое же количество техники. Задача — уничтожить! Всё просто. Раунд 1, как и все последующие этапы, 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 дополнительный набор 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 большим количеством юнитов. Также на карте появляются сооружения, которые ваша стратегия может захватывать, получая тем самым игровое преимущество на оппонентом. Дополнительно 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 дополнительный набор 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. Также в Финале вводится туман войны, ограничивающий видимость техники оппонента. Стратегии всегда доступны полные карты местности и погоды, а также информация обо всех сооружениях на карте. Радиус обзора юнитов достаточно большой. Таким образом, изменение

правил не должно сильно сказаться на локальном тактическом управлении. Однако для получения информации об отдалённых участках карты стратегии будет необходимо отправлять часть техники в разведку. Система проведения Финала имеет свои особенности. Этап по-прежнему делится на две части, однако они уже не будут состоять из волн. В каждой части этапа будут проведены игры между всеми парами участников Финала. Если позволит время и возможности тестирующей системы, операция 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.

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.

2.2 Описание игрового мира

Игровой мир является двумерным, а все юниты в нём имеют форму круга. 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. Игровая область ограничена квадратом, левый верхний угол которого имеет координаты (0.0, 0.0), а длина стороны равна 1024.0. Ни один юнит не может полностью или частично находиться за пределами игровой области.

В начале каждой игры техника первого игрока находится в левом верхнем углу игровой области, а техника второго игрока — в правом нижнем. При этом координаты для стратегии второго игрока передаются в преобразованном виде. Таким образом, стратегия всегда «думает», что начинает игру в левом верхнем углу карты, а противник находится в правом нижнем. Количество техники у каждого игрока в начале игры кратно 100. Техника разбита на группы по 100 юнитов в каждой. Вся техника в одной такой группе имеет одинаковый тип. Формацией группы является квадрат 10×10. Если в начале игры повернуть игровую область на 180° относительно её центра, то позиция каждого юнита второго игрока совпадёт с позицией юнита первого игрока до поворота. При этом юниты с совпадающей позицией будут иметь одинаковый тип.

Время в игре дискретное и измеряется в «тиках». В начале каждого тика симулятор игры передаёт стратегиям участников данные о состоянии мира, получает от них управляющие сигналы и обновляет состояние мира в соответствии с этими сигналами и ограничениями мира. Затем происходит расчёт изменения мира и объектов в нём за этот тик, и процесс повторяется снова с обновлёнными данными. Максимальная длительность любой игры равна 20000 тиков, однако игра может быть прекращена досрочно, если все юниты хотя бы одной стратегии были уничтожены либо все стратегии «упали». Крайне маловероятно, но всё-таки возможно, что все юниты обоих игроков будут уничтожены в один и тот же тик. Тогда дополнительные баллы получат все участники игры.

«Упавшая» стратегия больше не может управлять техникой. 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 10 секунд реального времени. Но в сумме на всю игру процессу стратегии выделяется

$$20 \times <\textit{длительность\_игры\_в\_тиках}> + 10000 \tag{2.1}$$

миллисекунд реального времени и

$$10 \times <\textit{длительность\_игры\_в\_тиках}> + 10000 \tag{2.2}$$

миллисекунд процессорного времени.[1] В формуле учитывается максимальная 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.

2.3 Типы техники

В мире CodeWars 2017 все юниты являются техникой. Существует 5 типов техники:

- **танк** (`TANK`): наземный юнит, эффективен против других наземных юнитов;
- **БМП** — боевая машина пехоты (`IFV`): наземный юнит, эффективен против воздушных юнитов;
- **ударный вертолёт** (`HELICOPTER`): воздушный юнит, эффективен против наземных юнитов;
- **истребитель** (`FIGHTER`): воздушный юнит, эффективен против других воздушных юнитов;
- **БРЭМ** — бронированная ремонтно-эвакуационная машина (`ARRV`): ремонтирует повреждённую технику.

Основными характеристиками техники являются текущее и максимальное значение прочности. При падении прочности до нуля юнит считается уничтоженным и убирается из игрового мира. Начальное и максимальное значение прочности каждого юнита равно 100. Все юниты являются кругами радиуса 2.0. Интервал между двумя последовательными атаками техники любого типа, кроме БРЭМ, составляет 60 тиков. БРЭМ не могут атаковать.

Сравнительные характеристики типов техники приведены в следующей таблице:

Характеристика \Тип техники	Танк	БМП	Вертолёт	Истребитель	БРЭМ
Скорость	0.4	0.6	1.2	2.0	0.6
Дальность обзора	80	80	100	120	60
Дальность атаки по наземным целям	20	18	20	—	—
Дальность атаки по воздушным целям	18	20	18	20	—
Урон одной атаки по наземным целям	100	90	100	—	—
Урон одной атаки по воздушным целям	60	80	80	100	—
Защита от атак наземных целей	80	60	40	70	50
Защита от атак воздушных целей	60	80	40	70	20

В отсутствие тумана войны, дальность обзора не оказывает никакого влияния на игру. При включенном тумане войны стратегия участника будет получать данные только о тех юнитах противника, которые находятся в пределах дальности[2] обзора хотя бы одного юнита этого участника.

Все атакующие юниты автоматически наносят урон, если в пределах дальности их атаки есть хотя бы один юнит противника, а также прошло достаточно времени с момента последней атаки. При наличии нескольких целей, для атаки выбирается случайная из них. При этом, чем выше урон одной атаки по конкретной цели, тем с большей вероятностью эта цель будет выбрана. Урон наносится мгновенно. Также считается, что все

[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.

[2] Здесь и далее под расстоянием между юнитами подразумевается расстояние между их центрами, если явно не указано другое.

юниты, совершающие атаку в один тик, делают это одновременно. Таким образом, дуэль двух одинаковых юнитов разных игроков закончится уничтожением обоих этих юнитов.

БРЭМ каждый тик автоматически ремонтируют на 0.1 одного из дружественных юнитов, прочность которых меньше максимальной, а расстояние до БРЭМ не превышает 10. При наличии нескольких целей, для ремонта выбирается случайная из них. При этом, чем ниже прочность этой цели, тем с большей вероятностью она будет выбрана. Значение скорости ремонта меньше единицы, поэтому на протяжении нескольких тиков может казаться, что прочность техники не восстанавливается, однако это не так. Суммарное восстановление прочности за прошедшие тики накапливается в специальном пуле. Техника считается уничтоженной, если целочисленная часть её прочности падает до нуля, независимо от значения в пуле.

2.4 Типы местности и погоды

Игровая область мира условно поделена на клетки размером 32.0×32.0. В каждой из таких клеток может быть один из трёх типов местности и один из трёх типов погоды. Тип местности влияет на различные параметры наземной техники; тип погоды, соответственно, — воздушной. Карты местности и погоды превращаются сами в себя, если их повернуть относительно центра игровой области на 180°. Обе карты не изменяются в процессе игры и всегда доступны стратегии, независимо от наличия тумана войны.

Характеристика \Тип местности	Равнина	Топь	Лес
Коэффициент скорости	1.0	0.6	0.8
Коэффициент дальности обзора	1.0	1.0	0.8
Коэффициент незаметности	1.0	1.0	0.6

Характеристика \Тип погоды	Ясно	Плотные облака	Сильный дождь
Коэффициент скорости	1.0	0.8	0.6
Коэффициент дальности обзора	1.0	0.8	0.6
Коэффициент незаметности	1.0	0.8	0.6

Если с коэффициентами скорости и дальности обзора всё очевидно, то коэффициент незаметности техники влияет на дальность обзора любого юнита противника при проверке видимости этой техники. Таким образом, юнит видит цель, если и только если расстояние до цели меньше или равно

$$<\textit{дальность\_обзора\_юнита}> \times <\textit{коэффициент\_дальности\_обзора\_юнита}> \times <\textit{коэффициент\_незаметности\_цели}> \tag{2.3}$$

Разумеется, при отключенном тумане войны значение имеет только коэффициент скорости.

2.5 Сооружения

Сооружения появляются в Раунде 2 турнира и представляют собой квадратные области на карте. Длина стороны каждого такого квадрата равна 64.0, а его левый верхний угол совпадает с левым верхним углом одной из клеток карты местности/погоды. Расположение сооружений симметрично для обоих игроков. Всего на карте может быть до 10 пар сооружений (являющихся отражениями друг друга). Стратегия получает информацию обо всех сооружениях, независимо от наличия тумана войны.

В начале игры все сооружения являются нейтральными. Стратегии могут захватывать сооружения, перемещая наземную технику в область сооружения. Каждый юнит в области сооружения вырабатывает 0.01 единицы захвата этого сооружения за игровой тик. При накоплении 100.0 единиц захвата процесс прекращается, а сооружение переходит под контроль стратегии. Если ваш оппонент полностью или частично захватил сооружение, сперва необходимо обнулить его уровень захвата. Обнуление захвата оппонента происходит таким же образом и с такой же скоростью, как и собственно сам захват. Если

оппонент уже контролирует сооружение, то он будет сохранять контроль над ним до тех пор, пока его уровень захвата не упадёт до нуля.

Типы сооружений:

- центр управления (`CONTROL_CENTER`): увеличивает лимит количества действий стратегии на 2 за 60 тиков;
- завод (`FACTORY`): каждые 60 тиков производит одну единицу техники, тип производимой техники определяется стратегией.

Новая техника стратегии появляется на заводе рядами слева направо, сверху вниз. Техника оппонента — также рядами, но справа налево, снизу вверх. Расстояние между центрами двух соседних юнитов, произведённых на заводе, равно 6.0. Если следующая по очереди позиция юнита занята, тогда она будет пропущена и так далее, пока не будет найдена свободная позиция. Если все позиции для производства юнитов заняты, завод приостановит производство новой техники.

2.6 Управление

В начале каждого тика симулятор игры отправляет стратегии сведения о текущем состоянии видимой части мира. В ответ стратегия отправляет набор инструкций (инкапсулированных в объекте класса `Move`) для управления техникой или просто пропускает ход. Изначально количество возможных действий стратегии ограничено 6-ю ходами за 60 тиков. Это значение может быть увеличено при захвате стратегией одного или нескольких центров управления. Действие стратегии будет проигнорировано игровым симулятором, если за последние $60 - 1$ тиков она уже совершила максимально доступное ей количество действий.

Инструкции стратегии обрабатываются в следующем порядке:

- Сперва происходит изменение мира в соответствии с пожеланиями стратегии: выполняются все действия по выделению юнитов, назначению юнитам групп, настраивается производство техники на заводе и т.д. Также обновляются приказы юнитов.
- Затем все юниты упорядочиваются случайным образом, и происходит их перемещение в соответствии с полученными или уже имеющимися приказами, а также с ограничением максимальной сокрости этих юнитов с учётом типа местности или погоды. В мире CodeWars 2017 нет инерции, а перемещение происходит мгновенно или вообще не происходит. Перемещение техники осуществляется последовательно, согласно выбранному порядку. При этом частичное перемещение техники не применяется. Если позицию техники невозможно изменить на полную величину вычисленного игровым симулятором перемещения[3], то её перемещение откладывается. После окончания перебора игровой симулятор снова итерируется по всем юнитам и пытается переместить тех, чья позиция в данный тик ещё не изменялась. Так происходит до тех пор, пока не будут перемещены все юниты. Если на очередной итерации не было перемещено ни одного юнита, то операция также прерывается.
- Затем все юниты, не находящиеся на перезарядке, одновременно совершают атакующие действия, а БРЭМ производят ремонт.
- В последнюю очередь изменяется уровень захвата сооружений и производится новая техника.

[3] Техника после перемещения частично или полностью находится за пределами карты либо пересекается с какой-либо другой техникой.

2.7 Столкновения юнитов

Коллизия юнитов между собой, а также с границами карты не допускается игровым симулятором. Исключение составляют воздушные юниты, принадлежащие разным игрокам.

2.8 Начисление баллов

Баллы начисляются за следующие дествия:

- За уничтожение юнита противника даётся 1 балл.
- Захват сооружения приносит стратегии 100 баллов.
- При уничтожении всех юнитов оппонента стратегия получает 1000 баллов. Игра при этом завершается.

Глава 3

Создание стратегии

3.1 Техническая часть

Сперва для создания стратегии вам необходимо выбрать один из ряда поддерживаемых языков программирования[4]: Java (Oracle JDK 8), C# (Roslyn 1.3+), C++14 (GNU MinGW C++ 6.2+), Python 2 (Python 2.7+), Python 3 (Python 3.5+), Pascal (Free Pascal 3.0+), Ruby (JRuby 9.1+, 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)[5]. 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[6]: 1 на 1, 1 на 1 с добавлением сооружений и 1 на 1 с добавлением сооружений и тумана войны. Для управления техникой 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 играх (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

---

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

[5] 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.

[6] Основными параметрами формата игры являются количество игроков, участвующих в нём, и количество юнитов, находящихся под управлением каждого игрока. Кратко формат записывается в виде `<количество_игроков>` × `<количество_юнитов>`, например запись 4 × 3 означает формат игры, в котором участвует 4 игрока, управляющих тремя юнитами каждый. В чемпионатах, все игры которых всегда проходят в формате дуэлей, может использоваться альтернативная форма записи, например, 1 игрок на 1 игрока (2 × `N` в каноническом виде), 1 юнит на 1 юнита (2 × 1) или 3 юнита на 3 юнита (2 × 3). Слова «игрок» и «юнит» в записи формата могут заменяться соответствующими иконками или вообще убираться, если из контекста понятно, о чём идёт речь. К формату игры может быть добавлено пояснение в случае, если краткая запись формата для разных этапов чемпионата совпадает.

любого игрока (в том числе и вашу собственную), созданную до момента вашей последней успешной посылки. Созданные вами игры не влияют на рейтинг.

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

- В течение двадцати минут нельзя отослать стратегию более трёх раз. Суммарный размер (без сжатия) стратегий, отправленных за двадцать минут, не может быть больше 3 Мб. Ограничение на размер одной посылки составляет 2 Мб.
- В течение двадцати минут нельзя создать более трёх пользовательских игр. После завершения каждого основного этапа соревнования это число автоматически увеличивается на единицу.

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 игры с её участием. Однако, после принятия данной посылки, система автоматически добавляет тестовую игру с двумя участниками (формат 1 на 1): непосредственно тестовой посылкой и стратегией из раздела «Быстрый старт». 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 CodeWars 2017 (раздел «Документация» → «Утилита Repeater») и разархивируйте. Для запуска Repeater вам необходимо установленное ПО Java 8+ 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. Утилита Repeater располагает только теми данными, которые отправлялись вашей стратегии, но не полной записью игры. Поэтому визуализация игры не осуществляется. 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 (*.bat для Windows или *.sh для *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 30 минут. По прошествии этого времени он посчитает стратегию «упавшей» и продолжит работу без неё.

3.2 Управление техникой

Для вашего игрока в начале игры создаётся объект класса `MyStrategy`, в полях которого стратегия может сохранять информацию о ходе игры. Управление техникой осуществляется с помощью метода `move` стратегии, который вызывается один раз за тик. Методу передаются следующие параметры:

- ваш игрок `me`;
- текущее состояние мира `world`;
- набор игровых констант `game`;
- объект `move`, устанавливая свойства которого, стратегия и управляет техникой.

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 `vehicle` 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 сперва выделяют всю вашу технику, а затем отправляют её на сближение с противником. Полную документацию по классам и методам 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(Player me, World world, Game game, Move move) {
        if (world.getTickIndex() == 0) {
            move.setAction(ActionType.CLEAR_AND_SELECT);
            move.setRight(world.getWidth());
            move.setBottom(world.getHeight());
            return;
        }

        if (world.getTickIndex() == 1) {
            move.setAction(ActionType.MOVE);
            move.setX(world.getWidth() / 2.0D);
            move.setY(world.getHeight() / 2.0D);
        }
    }
}
```

3.3.2 Пример для C#

```
using Com.CodeGame.CodeWars2017.DevKit.CSharpCgdk.Model;

namespace Com.CodeGame.CodeWars2017.DevKit.CSharpCgdk {
    public sealed class MyStrategy : IStrategy {
        public void Move(Player me, World world, Game game, Move move) {
            if (world.TickIndex == 0) {
                move.Action = ActionType.ClearAndSelect;
                move.Right = world.Width;
                move.Bottom = world.Height;
                return;
            }

            if (world.TickIndex == 1) {
                move.Action = ActionType.Move;
                move.X = world.Width / 2.0D;
                move.Y = world.Height / 2.0D;
            }
        }
    }
}
```

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 Player& me, const World& world, const Game& game, Move& move) {
    if (world.getTickIndex() == 0) {
        move.setAction(ACTION_CLEAR_AND_SELECT);
        move.setRight(world.getWidth());
        move.setBottom(world.getHeight());
        return;
    }

    if (world.getTickIndex() == 1) {
        move.setAction(ACTION_MOVE);
        move.setX(world.getWidth() / 2.0);
        move.setY(world.getHeight() / 2.0);
    }
}

MyStrategy::MyStrategy() { }
```

3.3.4 Пример для Python 2

```
from model.ActionType import ActionType
from model.Game import Game
from model.Move import Move
from model.Player import Player
from model.World import World


class MyStrategy:
    def move(self, me, world, game, move):
        """
        @type me: Player
        @type world: World
        @type game: Game
        @type move: Move
        """
        if world.tick_index == 0:
            move.action = ActionType.CLEAR_AND_SELECT
            move.right = world.width
            move.bottom = world.height

        if world.tick_index == 1:
            move.action = ActionType.MOVE
            move.x = world.width / 2.0
            move.y = world.height / 2.0
```

3.3.5 Пример для Python 3

```
from model.ActionType import ActionType
from model.Game import Game
from model.Move import Move
from model.Player import Player
from model.World import World


class MyStrategy:
    def move(self, me: Player, world: World, game: Game, move: Move):
        if world.tick_index == 0:
            move.action = ActionType.CLEAR_AND_SELECT
            move.right = world.width
            move.bottom = world.height

        if world.tick_index == 1:
            move.action = ActionType.MOVE
            move.x = world.width / 2.0
            move.y = world.height / 2.0
```

3.3.6 Пример для Pascal

```
unit MyStrategy;

interface

uses
  StrategyControl, TypeControl, ActionTypeControl, CircularUnitControl, FacilityControl,
  FacilityTypeControl, GameControl, MoveControl, PlayerContextControl, PlayerControl,
  TerrainTypeControl, UnitControl, VehicleControl, VehicleTypeControl, VehicleUpdateControl,
  WeatherTypeControl, WorldControl;

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

  end;

implementation

uses
  Math;

procedure TMyStrategy.Move(me: TPlayer; world: TWorld; game: TGame; move: TMove);
begin
  if world.TickIndex = 0 then begin
    move.Action := ACTION_CLEAR_AND_SELECT;
    move.Right := world.Width;
    move.Bottom := world.Height;
    exit;
  end;

  if world.TickIndex = 1 then begin
    move.Action := ACTION_MOVE;
    move.X := world.Width / 2.0;
    move.Y := world.Height / 2.0;
  end;
end;

end.
```

3.3.7 Пример для Ruby

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

class MyStrategy
  # @param [Player] me
  # @param [World] world
  # @param [Game] game
  # @param [Move] move
  def move(me, world, game, move)
    if world.tick_index == 0
      move.action = ActionType::CLEAR_AND_SELECT
      move.right = world.width
      move.bottom = world.height
    end

    if world.tick_index == 1
      move.action = ActionType::MOVE
      move.x = world.width / 2.0
      move.y = world.height / 2.0
    end
  end
end
```

Глава 4

Package model

Package Contents *Page*

4.1 Classes

4.1.1 CLASS **ActionType**

Возможные действия игрока.

Игрок не может совершить новое действие, если в течение последних `game.actionDetectionInterval - 1` игровых тиков он уже совершил максимально возможное для него количество действий. В начале игры это ограничение для каждого игрока равно `game.baseActionCount`. Ограничение увеличивается за каждый контролируемый игроком центр управления (`FacilityType.CONTROL_CENTER`).

Большинство действий требует указания дополнительных параметров, являющихся полями объекта `move`. В случае, если эти параметры установлены некорректно либо указаны не все обязательные параметры, действие будет проигнорировано игровым симулятором. Любое действие, отличное от `NONE`, даже проигнорированное, будет учтено в счётчике действий игрока.

DECLARATION

public final class ActionType

extends Enum

FIELDS

- public static final ActionType NONE
  - Ничего не делать.
- public static final ActionType CLEAR_AND_SELECT
  - Пометить юнитов, соответствующих некоторым параметрам, как выделенных. При этом, со всех остальных юнитов выделение снимается. Юниты других игроков автоматически исключаются из выделения.
- public static final ActionType ADD_TO_SELECTION
  - Пометить юнитов, соответствующих некоторым параметрам, как выделенных. При этом, выделенные ранее юниты остаются выделенными. Юниты других игроков автоматически исключаются из выделения.
- public static final ActionType DESELECT
  - Снять выделение с юнитов, соответствующих некоторым параметрам.
- public static final ActionType ASSIGN
  - Установить для выделенных юнитов принадлежность к группе.
- public static final ActionType DISMISS
  - Убрать у выделенных юнитов принадлежность к группе.
- public static final ActionType DISBAND

- Расформировать группу.

- public static final ActionType MOVE
  - Приказать выделенным юнитам меремещаться в указанном направлении.
- public static final ActionType ROTATE
  - Приказать выделенным юнитам поворачиваться относительно некоторой точки.
- public static final ActionType SETUP_VEHICLE_PRODUCTION
  - Настроить производство нужного типа техники на заводе (`FacilityType.VEHICLE_FACTORY`).

4.1.2 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.

4.1.3 CLASS **Facility**

Класс, определяющий сооружение — прямоугольную область на карте.

DECLARATION

public class Facility **extends** Object

- *getCapturePoints*
`public double` **getCapturePoints**`( )`
  - **Returns** - Возвращает индикатор захвата сооружения в интервале от `-game.maxFacilityCapturePoints` до `game.maxFacilityCapturePoints`. Если индикатор находится в положительной зоне, очки захвата принадлежат вам, иначе вашему противнику.

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

- *getLeft*
`public double` **getLeft**`( )`
  - **Returns** - Возвращает абсциссу левой границы сооружения.

- *getOwnerPlayerId*
`public long` **getOwnerPlayerId**`( )`
  - **Returns** - Возвращает идентификатор игрока, захватившего сооружение, или `-1`, если сооружение никем не контролируется.

- *getProductionProgress*
`public int` **getProductionProgress**`( )`
  - **Returns** - Возвращает неотрицательное число — прогресс производства техники. Применимо только к заводу (`FacilityType.VEHICLE_FACTORY`).

- *getTop*
`public double` **getTop**`( )`
  - **Returns** - Возвращает ординату верхней границы сооружения.

- *getType*
`public FacilityType` **getType**`( )`
  - **Returns** - Возвращает тип сооружения.

- *getVehicleType*
`public VehicleType` **getVehicleType**`( )`
  - **Returns** - Возвращает тип техники, производящейся в данном сооружении, или `null`. Применимо только к заводу (`FacilityType.VEHICLE_FACTORY`).

4.1.4 CLASS **FacilityType**

Тип сооружения.

DECLARATION

public final class FacilityType

extends Enum

FIELDS

- public static final FacilityType CONTROL_CENTER
  - Центр управления. Увеличивает возможное количество действий игрока на `game.additionalActionCountPerControlCenter` за `game.actionDetectionInterval` игровых тиков.
- public static final FacilityType VEHICLE_FACTORY
  - Завод. Может производить технику любого типа по выбору игрока.

4.1.5 CLASS **Game**

Предоставляет доступ к различным игровым константам.

DECLARATION

public class Game

extends Object

METHODS

- *getActionDetectionInterval*
  `public int` **getActionDetectionInterval**`( )`
  - **Returns** - Возвращает интервал, учитываемый в ограничении количества действий стратегии.
- *getAdditionalActionCountPerControlCenter*
  `public int` **getAdditionalActionCountPerControlCenter**`( )`
  - **Returns** - Возвращает дополнительное количество действий за каждый захваченный центр управления (`FacilityType.CONTROL_CENTER`).
- *getArrvAerialDefence*
  `public int` **getArrvAerialDefence**`( )`

- **Returns** - Возвращает защиту БРЭМ от атак воздушной техники.

- *getArrvDurability*
`public int` **getArrvDurability**`( )`

  - **Returns** - Возвращает максимальную прочность БРЭМ.

- *getArrvGroundDefence*
`public int` **getArrvGroundDefence**`( )`

  - **Returns** - Возвращает защиту БРЭМ от атак наземной техники.

- *getArrvProductionCost*
`public int` **getArrvProductionCost**`( )`

  - **Returns** - Возвращает количество тиков, необхожимое для производства одной БРЭМ на заводе (`FacilityType.VEHICLE_FACTORY`).

- *getArrvRepairRange*
`public double` **getArrvRepairRange**`( )`

  - **Returns** - Возвращает максимальное расстояние (от центра до центра), на котором БРЭМ может ремонтировать дружественную технику.

- *getArrvRepairSpeed*
`public double` **getArrvRepairSpeed**`( )`

  - **Returns** - Возвращает максимальное количество прочности, которое БРЭМ может восстановить дружественной технике за один тик.

- *getArrvSpeed*
`public double` **getArrvSpeed**`( )`

  - **Returns** - Возвращает максимальную скорость БРЭМ.

- *getArrvVisionRange*
`public double` **getArrvVisionRange**`( )`

  - **Returns** - Возвращает базовый радиус обзора БРЭМ.

- *getBaseActionCount*
`public int` **getBaseActionCount**`( )`

  - **Returns** - Возвращает базовое количество действий, которое может совершить стратегия за `actionDetectionInterval` последовательных тиков.

- *getClearWeatherSpeedFactor*
`public double` **getClearWeatherSpeedFactor**`( )`

  - **Returns** - Возвращает мультипликатор максимальной скорости воздушной техники, находящейся в области ясной погоды (`WeatherType.CLEAR`).

- *getClearWeatherStealthFactor*
`public double` **getClearWeatherStealthFactor**`( )`

  - **Returns** - Возвращает мультипликатор радиуса обзора любой техники при обнаружении воздушной техники противника, находящейся в области ясной погоды (`WeatherType.CLEAR`).

- *getClearWeatherVisionFactor*
`public double` **getClearWeatherVisionFactor**`( )`

  - **Returns** - Возвращает мультипликатор радиуса обзора воздушной техники, находящейся в области ясной погоды (`WeatherType.CLEAR`).

- *getCloudWeatherSpeedFactor*
`public double` **getCloudWeatherSpeedFactor**`( )`

  - **Returns** - Возвращает мультипликатор максимальной скорости воздушной техники, находящейся в плотных облаках (`WeatherType.CLOUD`).

- *getCloudWeatherStealthFactor*
`public double` **getCloudWeatherStealthFactor**`( )`

  - **Returns** - Возвращает мультипликатор радиуса обзора любой техники при обнаружении воздушной техники противника, находящейся в плотных облаках (`WeatherType.CLOUD`).

- *getCloudWeatherVisionFactor*
`public double` **getCloudWeatherVisionFactor**`( )`

  - **Returns** - Возвращает мультипликатор радиуса обзора воздушной техники, находящейся в плотных облаках (`WeatherType.CLOUD`).

- *getFacilityCapturePointsPerVehiclePerTick*
`public double` **getFacilityCapturePointsPerVehiclePerTick**`( )`

  - **Returns** - Возвращает скорость изменения индикатора захвата сооружения (`facility.capturePoints`) за каждую единицу техники, центр которой находится внутри сооружения.

- *getFacilityCaptureScore*
`public int` **getFacilityCaptureScore**`( )`

  - **Returns** - Возвращает количество баллов за захват сооружения.

- *getFacilityHeight*
`public double` **getFacilityHeight**`( )`

  - **Returns** - Возвращает высоту сооружения.

- *getFacilityWidth*
`public double` **getFacilityWidth**`( )`

  - **Returns** - Возвращает ширину сооружения.

- *getFighterAerialAttackRange*
`public double` **getFighterAerialAttackRange**`( )`

  - **Returns** - Возвращает дальность атаки истребителя по воздушным целям.

- *getFighterAerialDamage*
`public int` **getFighterAerialDamage**`( )`

  - **Returns** - Возвращает урон одной атаки истребителя по воздушной технике.

- *getFighterAerialDefence*
`public int` **getFighterAerialDefence**`( )`

  - **Returns** - Возвращает защиту истребителя от атак воздушной техники.

- *getFighterAttackCooldownTicks*
`public int` **getFighterAttackCooldownTicks**`( )`

  - **Returns** - Возвращает интервал в тиках между двумя последовательными атаками истребителя.

- *getFighterDurability*
`public int` **getFighterDurability**`( )`

  - **Returns** - Возвращает максимальную прочность истребителя.

- *getFighterGroundAttackRange*
`public double` **getFighterGroundAttackRange**`( )`

  - **Returns** - Возвращает дальность атаки истребителя по наземным целям.

- *getFighterGroundDamage*
`public int` **getFighterGroundDamage**`( )`

  - **Returns** - Возвращает урон одной атаки истребителя по наземной технике.

- *getFighterGroundDefence*
`public int` **getFighterGroundDefence**`( )`

  - **Returns** - Возвращает защиту истребителя от атак наземной техники.

- *getFighterProductionCost*
`public int` **getFighterProductionCost**`( )`

  - **Returns** - Возвращает количество тиков, необхожимое для производства одного истребителя на заводе (`FacilityType.VEHICLE_FACTORY`).

- *getFighterSpeed*
`public double` **getFighterSpeed**`( )`

  - **Returns** - Возвращает максимальную скорость истребителя.

- *getFighterVisionRange*
`public double` **getFighterVisionRange**`( )`

  - **Returns** - Возвращает базовый радиус обзора истребителя.

- *getForestTerrainSpeedFactor*
`public double` **getForestTerrainSpeedFactor**`( )`

  - **Returns** - Возвращает мультипликатор максимальной скорости наземной техники, находящейся в лесистой местности (`TerrainType.FOREST`).

- *getForestTerrainStealthFactor*
`public double` **getForestTerrainStealthFactor**`( )`

  - **Returns** - Возвращает мультипликатор радиуса обзора любой техники при обнаружении наземной техники противника, находящейся в лесистой местности (`TerrainType.FOREST`).

- *getForestTerrainVisionFactor*
`public double` **getForestTerrainVisionFactor**`( )`

  - **Returns** - Возвращает мультипликатор радиуса обзора наземной техники, находящейся в лесистой местности (`TerrainType.FOREST`).

- *getHelicopterAerialAttackRange*
`public double` **getHelicopterAerialAttackRange**`( )`

  - **Returns** - Возвращает дальность атаки ударного вертолёта по воздушным целям.

- *getHelicopterAerialDamage*
`public int` **getHelicopterAerialDamage**`( )`

  - **Returns** - Возвращает урон одной атаки ударного вертолёта по воздушной технике.

- *getHelicopterAerialDefence*
`public int` **getHelicopterAerialDefence**`( )`

  - **Returns** - Возвращает защиту ударного вертолёта от атак воздушной техники.

- *getHelicopterAttackCooldownTicks*
`public int` **getHelicopterAttackCooldownTicks**`( )`

  - **Returns** - Возвращает интервал в тиках между двумя последовательными атаками ударного вертолёта.

- *getHelicopterDurability*
`public int` **getHelicopterDurability**`( )`

  - **Returns** - Возвращает максимальную прочность ударного вертолёта.

- *getHelicopterGroundAttackRange*
`public double` **getHelicopterGroundAttackRange**`( )`

  - **Returns** - Возвращает дальность атаки ударного вертолёта по наземным целям.

- *getHelicopterGroundDamage*
`public int` **getHelicopterGroundDamage**`( )`

  - **Returns** - Возвращает урон одной атаки ударного вертолёта по наземной технике.

- *getHelicopterGroundDefence*
`public int` **getHelicopterGroundDefence**`( )`

  - **Returns** - Возвращает защиту ударного вертолёта от атак наземной техники.

- *getHelicopterProductionCost*
`public int` **getHelicopterProductionCost**`( )`

  - **Returns** - Возвращает количество тиков, необхожимое для производства одного ударного вертолёта на заводе (`FacilityType.VEHICLE_FACTORY`).

- *getHelicopterSpeed*
`public double` **getHelicopterSpeed**`( )`

  - **Returns** - Возвращает максимальную скорость ударного вертолёта.

- *getHelicopterVisionRange*
`public double` **getHelicopterVisionRange**`( )`

  - **Returns** - Возвращает базовый радиус обзора ударного вертолёта.

- *getIfvAerialAttackRange*
`public double` **getIfvAerialAttackRange**`( )`

  - **Returns** - Возвращает дальность атаки БМП по воздушным целям.

- *getIfvAerialDamage*
`public int` **getIfvAerialDamage**`( )`

  - **Returns** - Возвращает урон одной атаки БМП по воздушной технике.

- *getIfvAerialDefence*
`public int` **getIfvAerialDefence**`( )`

  - **Returns** - Возвращает защиту БМП от атак воздушной техники.

- *getIfvAttackCooldownTicks*
`public int` **getIfvAttackCooldownTicks**`( )`

  - **Returns** - Возвращает интервал в тиках между двумя последовательными атаками БМП.

- *getIfvDurability*
`public int` **getIfvDurability**`( )`
  - **Returns** - Возвращает максимальную прочность БМП.

- *getIfvGroundAttackRange*
`public double` **getIfvGroundAttackRange**`( )`
  - **Returns** - Возвращает дальность атаки БМП по наземным целям.

- *getIfvGroundDamage*
`public int` **getIfvGroundDamage**`( )`
  - **Returns** - Возвращает урон одной атаки БМП по наземной технике.

- *getIfvGroundDefence*
`public int` **getIfvGroundDefence**`( )`
  - **Returns** - Возвращает защиту БМП от атак наземной техники.

- *getIfvProductionCost*
`public int` **getIfvProductionCost**`( )`
  - **Returns** - Возвращает количество тиков, необхожимое для производства одной БМП на заводе (`FacilityType.VEHICLE_FACTORY`).

- *getIfvSpeed*
`public double` **getIfvSpeed**`( )`
  - **Returns** - Возвращает максимальную скорость БМП.

- *getIfvVisionRange*
`public double` **getIfvVisionRange**`( )`
  - **Returns** - Возвращает базовый радиус обзора БМП.

- *getMaxFacilityCapturePoints*
`public double` **getMaxFacilityCapturePoints**`( )`
  - **Returns** - Возвращает максимально возможную абсолютную величину индикатора захвата сооружения (`facility.capturePoints`).

- *getMaxUnitGroup*
`public int` **getMaxUnitGroup**`( )`
  - **Returns** - Возвращает максимально возможный индекс группы юнитов.

- *getPlainTerrainSpeedFactor*
`public double` **getPlainTerrainSpeedFactor**`( )`
  - **Returns** - Возвращает мультипликатор максимальной скорости наземной техники, находящейся на равнинной местности (`TerrainType.PLAIN`).

- *getPlainTerrainStealthFactor*
`public double` **getPlainTerrainStealthFactor**`( )`
  - **Returns** - Возвращает мультипликатор радиуса обзора любой техники при обнаружении наземной техники противника, находящейся на равнинной местности (`TerrainType.PLAIN`).

- *getPlainTerrainVisionFactor*
`public double` **getPlainTerrainVisionFactor**`( )`
  - **Returns** - Возвращает мультипликатор радиуса обзора наземной техники, находящейся на равнинной местности (`TerrainType.PLAIN`).

- *getRainWeatherSpeedFactor*
`public double` **getRainWeatherSpeedFactor**`( )`
  - **Returns** - Возвращает мультипликатор максимальной скорости воздушной техники, находящейся в условиях сильного дождя (`WeatherType.RAIN`).

- *getRainWeatherStealthFactor*
`public double` **getRainWeatherStealthFactor**`( )`
  - **Returns** - Возвращает мультипликатор радиуса обзора любой техники при обнаружении воздушной техники противника, находящейся в условиях сильного дождя (`WeatherType.RAIN`).

- *getRainWeatherVisionFactor*
`public double` **getRainWeatherVisionFactor**`( )`
  - **Returns** - Возвращает мультипликатор радиуса обзора воздушной техники, находящейся в условиях сильного дождя (`WeatherType.RAIN`).

- *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.

- *getSwampTerrainSpeedFactor*
`public double` **getSwampTerrainSpeedFactor**`( )`
  - **Returns** - Возвращает мультипликатор максимальной скорости наземной техники, находящейся в болотистой местности (`TerrainType.SWAMP`).

- *getSwampTerrainStealthFactor*
`public double` **getSwampTerrainStealthFactor**`( )`
  - **Returns** - Возвращает мультипликатор радиуса обзора любой техники при обнаружении наземной техники противника, находящейся в болотистой местности (`TerrainType.SWAMP`).

- *getSwampTerrainVisionFactor*
`public double` **getSwampTerrainVisionFactor**`( )`
  - **Returns** - Возвращает мультипликатор радиуса обзора наземной техники, находящейся в болотистой местности (`TerrainType.SWAMP`).

- *getTankAerialAttackRange*
`public double` **getTankAerialAttackRange**`( )`
  - **Returns** - Возвращает дальность атаки танка по воздушным целям.

- *getTankAerialDamage*
`public int` **getTankAerialDamage**`( )`
  - **Returns** - Возвращает урон одной атаки танка по воздушной технике.

- *getTankAerialDefence*
`public int` **getTankAerialDefence**`( )`
  - **Returns** - Возвращает защиту танка от атак воздушной техники.

- *getTankAttackCooldownTicks*
`public int` **getTankAttackCooldownTicks**`( )`
  - **Returns** - Возвращает интервал в тиках между двумя последовательными атаками танка.

- *getTankDurability*
`public int` **getTankDurability**`( )`

  — **Returns** - Возвращает максимальную прочность танка.

- *getTankGroundAttackRange*
`public double` **getTankGroundAttackRange**`( )`

  — **Returns** - Возвращает дальность атаки танка по наземным целям.

- *getTankGroundDamage*
`public int` **getTankGroundDamage**`( )`

  — **Returns** - Возвращает урон одной атаки танка по наземной технике.

- *getTankGroundDefence*
`public int` **getTankGroundDefence**`( )`

  — **Returns** - Возвращает защиту танка от атак наземной техники.

- *getTankProductionCost*
`public int` **getTankProductionCost**`( )`

  — **Returns** - Возвращает количество тиков, необхожимое для производства одного танка на заводе (`FacilityType.VEHICLE_FACTORY`).

- *getTankSpeed*
`public double` **getTankSpeed**`( )`

  — **Returns** - Возвращает максимальную скорость танка.

- *getTankVisionRange*
`public double` **getTankVisionRange**`( )`

  — **Returns** - Возвращает базовый радиус обзора танка.

- *getTerrainWeatherMapColumnCount*
`public int` **getTerrainWeatherMapColumnCount**`( )`

  — **Returns** - Возвращает количество столбцов в картах местности и погоды.

- *getTerrainWeatherMapRowCount*
`public int` **getTerrainWeatherMapRowCount**`( )`

  — **Returns** - Возвращает количество строк в картах местности и погоды.

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

  — **Returns** - Возвращает базовую длительность игры в тиках. Реальная длительность может отличаться от этого значения в меньшую сторону. Эквивалентно `world.tickCount`.

- *getVehicleEliminationScore*
`public int` **getVehicleEliminationScore**`( )`

  — **Returns** - Возвращает количество баллов за уничтожение юнита противника.

- *getVehicleRadius*
`public double` **getVehicleRadius**`( )`

  — **Returns** - Возвращает радиус техники.

- *getVictoryScore*
`public int` **getVictoryScore**`( )`
  - **Returns** - Возвращает количество баллов, получаемое игроком в случае уничтожения всех юнитов противника.

- *getWorldHeight*
`public double` **getWorldHeight**`( )`
  - **Returns** - Возвращает высоту карты.

- *getWorldWidth*
`public double` **getWorldWidth**`( )`
  - **Returns** - Возвращает ширину карты.

- *isFogOfWarEnabled*
`public boolean` **isFogOfWarEnabled**`( )`
  - **Returns** - Возвращает `true`, если и только если в данной игре включен режим частичной видимости.

4.1.6 CLASS **Move**

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

DECLARATION

---

public class Move **extends** Object

METHODS

- *getAction*
`public ActionType` **getAction**`( )`
  - **Returns** - Возвращает текущее действие игрока.

- *getAngle*
`public double` **getAngle**`( )`
  - **Returns** - Возвращает текущий угол поворота.

- *getBottom*
`public double` **getBottom**`( )`
  - **Returns** - Возвращает текущую нижнюю границу прямоугольной рамки, предназначенной для выделения юнитов.

- *getFacilityId*
`public long` **getFacilityId**`( )`
  - **Returns** - Возвращает текущий идентификатор сооружения.

- *getGroup*
`public int` **getGroup**`( )`
  - **Returns** - Возвращает текущую группу юнитов.

- *getLeft*
`public double` **getLeft**`( )`
  - **Returns** - Возвращает текущую левую границу прямоугольной рамки, предназначенной для выделения юнитов.

- *getMaxAngularSpeed*
`public double` **getMaxAngularSpeed**`( )`
  - **Returns** - Возвращает текущее абсолютное ограничение скорости поворота.

- *getMaxSpeed*
`public double` **getMaxSpeed**`( )`
  - **Returns** - Возвращает текущее ограничение линейной скорости.

- *getRight*
`public double` **getRight**`( )`
  - **Returns** - Возвращает текущую правую границу прямоугольной рамки, предназначенной для выделения юнитов.

- *getTop*
`public double` **getTop**`( )`
  - **Returns** - Возвращает текущую верхнюю границу прямоугольной рамки, предназначенной для выделения юнитов.

- *getVehicleType*
`public VehicleType` **getVehicleType**`( )`
  - **Returns** - Возвращает текущий тип техники.

- *getX*
`public double` **getX**`( )`
  - **Returns** - Возвращает текущую абсциссу точки или вектора.

- *getY*
`public double` **getY**`( )`
  - **Returns** - Возвращает текущую ординату точки или вектора.

- *setAction*
`public void` **setAction**`( ActionType` **action** `)`
  - **Usage**
    - Устанавливает действие игрока.

- *setAngle*
`public void` **setAngle**`( double` **angle** `)`
  - **Usage**

* Задаёт угол поворота.

  Является обязательным параметром для действия `ActionType.ROTATE` и задаёт угол поворота относительно точки (`x`, `y`). Положительные значения соответствуют повороту по часовой стрелке.

  Корректными значениями являются вещественные числа от `-PI` до `PI` включительно.

- *setBottom*
  `public void` **setBottom**`( double` **bottom** `)`
  - **Returns** - Устанавливает нижнюю границу прямоугольной рамки для выделения юнитов.

    Является обязательным параметром для действий `ActionType.CLEAR_AND_SELECT`, `ActionType.ADD_TO_SELECTION` и `ActionType.DESELECT`, если не установлена группа юнитов. В противном случае граница будет проигнорирована.

    Корректными значениями являются вещественные числа от `top` до `game.worldHeight` включительно.

- *setFacilityId*
  `public void` **setFacilityId**`( long` **facilityId** `)`
  - **Usage**
    * Устанавливает идентификатор сооружения.

      Является обязательным параметром для действия `ActionType.SETUP_VEHICLE_PRODUCTION`. Если сооружение с данным идентификатором отсутствует в игре, не является заводом по производству техники (`FacilityType.VEHICLE_FACTORY`) или принадлежит другому игроку, то действие будет проигнорировано.

- *setGroup*
  `public void` **setGroup**`( int` **group** `)`
  - **Usage**
    * Устанавливает группу юнитов для различных действий.

      Является опциональным параметром для действий `ActionType.CLEAR_AND_SELECT`, `ActionType.ADD_TO_SELECTION` и `ActionType.DESELECT`. Если для этих действий группа юнитов установлена, то параметр `vehicleType`, а также параметры прямоугольной рамки `left`, `top`, `right` и `bottom` будут проигнорированы.

      Является обязательным параметром для действий `ActionType.ASSIGN`, `ActionType.DISMISS` и `ActionType.DISBAND`. Для действия `ActionType.DISBAND` является единственным учитываемым параметром.

      Корректными значениями являются целые числа от `1` до `game.maxUnitGroup` включительно.

- *setLeft*
  `public void` **setLeft**`( double` **left** `)`
  - **Returns** - Устанавливает левую границу прямоугольной рамки для выделения юнитов.

    Является обязательным параметром для действий `ActionType.CLEAR_AND_SELECT`, `ActionType.ADD_TO_SELECTION` и `ActionType.DESELECT`, если не установлена группа юнитов. В противном случае граница будет проигнорирована.

    Корректными значениями являются вещественные числа от `0.0` до `right` включительно.

- *setMaxAngularSpeed*
`public void` **setMaxAngularSpeed**`( double` **maxAngularSpeed** `)`
  - **Usage**
    * Устанавливает абсолютное ограничение скорости поворота в радианах за тик.

      Является опциональным параметром для действия `ActionType.ROTATE`. Если для этого действия установлено ограничение скорости поворота, то параметр `maxSpeed` будет проигнорирован.

      Корректными значениями являются вещественные числа в интервале от `0.0` до `PI` включительно. При этом, `0.0` означает, что ограничение отсутствует.

- *setMaxSpeed*
`public void` **setMaxSpeed**`( double` **maxSpeed** `)`
  - **Usage**
    * Устанавливает абсолютное ограничение линейной скорости.

      Является опциональным параметром для действий `ActionType.MOVE` и `ActionType.ROTATE`. Если для действия `ActionType.ROTATE` установлено ограничение скорости поворота, то этот параметр будет проигнорирован.

      Корректными значениями являются вещественные неотрицательные числа. При этом, `0.0` означает, что ограничение отсутствует.

- *setRight*
`public void` **setRight**`( double` **right** `)`
  - **Returns** - Устанавливает правую границу прямоугольной рамки для выделения юнитов.

    Является обязательным параметром для действий `ActionType.CLEAR_AND_SELECT`, `ActionType.ADD_TO_SELECTION` и `ActionType.DESELECT`, если не установлена группа юнитов. В противном случае граница будет проигнорирована.

    Корректными значениями являются вещественные числа от `left` до `game.worldWidth` включительно.

- *setTop*
`public void` **setTop**`( double` **top** `)`
  - **Returns** - Устанавливает верхнюю границу прямоугольной рамки для выделения юнитов.

    Является обязательным параметром для действий `ActionType.CLEAR_AND_SELECT`, `ActionType.ADD_TO_SELECTION` и `ActionType.DESELECT`, если не установлена группа юнитов. В противном случае граница будет проигнорирована.

    Корректными значениями являются вещественные числа от `0.0` до `bottom` включительно.

- *setVehicleType*
`public void` **setVehicleType**`( VehicleType` **vehicleType** `)`
  - **Usage**
    * Устанавливает тип техники.

      Является опциональным параметром для действий `ActionType.CLEAR_AND_SELECT`, `ActionType.ADD_TO_SELECTION` и `ActionType.DESELECT`. Указанные действия будут применены только к технике выбранного типа. Параметр будет проигнорирован, если

установлена группа юнитов.

Является опциональным параметром для действия `ActionType.SETUP_VEHICLE_PRODUCTION`. Завод будет настроен на производство техники данного типа. При этом, прогресс производства будет обнулён. Если данный параметр не установлен, то производство техники на заводе будет остановлено.

- *setX*
`public void setX( double  x )`
  - **Usage**
    * Устанавливает абсциссу точки или вектора.

      Является обязательным параметром для действия `ActionType.MOVE` и задаёт целевую величину смещения юнитов вдоль оси абсцисс.

      Является обязательным параметром для действия `ActionType.ROTATE` и задаёт абсциссу точки, относительно которой необходимо совершить поворот.

      Корректными значениями для действия `ActionType.MOVE` являются вещественные числа от `-game.worldWidth` до `game.worldWidth` включительно. Корректными значениями для действия `ActionType.ROTATE` являются вещественные числа от `-game.worldWidth` до `2.0 * game.worldWidth` включительно.

- *setY*
`public void setY( double  y )`
  - **Usage**
    * Устанавливает ординату точки или вектора.

      Является обязательным параметром для действия `ActionType.MOVE` и задаёт целевую величину смещения юнитов вдоль оси ординат.

      Является обязательным параметром для действия `ActionType.ROTATE` и задаёт ординату точки, относительно которой необходимо совершить поворот.

      Корректными значениями для действия `ActionType.MOVE` являются вещественные числа от `-game.worldHeight` до `game.worldHeight` включительно. Корректными значениями для действия `ActionType.ROTATE` являются вещественные числа от `-game.worldHeight` до `2.0 * game.worldHeight` включительно.

4.1.7 CLASS **Player**

Содержит данные о текущем состоянии игрока.

DECLARATION

---

public class Player **extends** Object

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

- *getRemainingActionCooldownTicks*
  `public int` **getRemainingActionCooldownTicks**`( )`
  - **Returns** - Возвращает количество тиков, оставшееся до любого следующего действия. Если значение равно `0`, игрок может совершить действие в данный тик.

- *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.

4.1.8 CLASS **TerrainType**

Тип местности.

DECLARATION

public final class TerrainType

extends Enum

FIELDS

- public static final TerrainType PLAIN
  - Равнина.
- public static final TerrainType SWAMP
  - Топь.
- public static final TerrainType FOREST
  - Лес.

4.1.9 CLASS **Unit**

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

DECLARATION

public abstract class Unit

extends Object

METHODS

- *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.

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

- *getSquaredDistanceTo*
  `public double` **getSquaredDistanceTo**`( Unit  unit )`
  - **Parameters**
    - `unit` - Объект, до центра которого необходимо определить квадрат расстояния.
  - **Returns** - Возвращает квадрат расстояния от центра данного объекта до центра указанного объекта.

- *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.

4.1.10 CLASS **Vehicle**

Класс, определяющий технику. Содержит также все свойства круглых объектов.

DECLARATION

public class Vehicle **extends** CircularUnit

METHODS

- *getAerialAttackRange*
`public double` **getAerialAttackRange**`( )`
  - **Returns** - Возвращает максимальное расстояние (от центра до центра), на котором данная техника может атаковать воздушные объекты.

- *getAerialDamage*
`public int` **getAerialDamage**`( )`
  - **Returns** - Возвращает урон одной атаки по воздушному объекту.

- *getAerialDefence*
`public int` **getAerialDefence**`( )`
  - **Returns** - Возвращает защиту от атак воздушых юнитов.

- *getAttackCooldownTicks*
`public int` **getAttackCooldownTicks**`( )`
  - **Returns** - Возвращает минимально возможный интервал между двумя последовательными атаками данной техники.

- *getDurability*
`public int` **getDurability**`( )`
  - **Returns** - Возвращает текущую прочность.

- *getGroundAttackRange*
`public double` **getGroundAttackRange**`( )`
  - **Returns** - Возвращает максимальное расстояние (от центра до центра), на котором данная техника может атаковать наземные объекты.

- *getGroundDamage*
`public int` **getGroundDamage**`( )`
  - **Returns** - Возвращает урон одной атаки по наземному объекту.

- *getGroundDefence*
`public int` **getGroundDefence**`( )`
  - **Returns** - Возвращает защиту от атак наземных юнитов.

- *getGroups*
`public int[]` **getGroups**`( )`
  - **Returns** - Возвращает группы, в которые входит эта техника.

- *getMaxDurability*
`public int` **getMaxDurability**`( )`
  - **Returns** - Возвращает максимальную прочность.

- *getMaxSpeed*
`public double` **getMaxSpeed**`( )`
  - **Returns** - Возвращает максимальное расстояние, на которое данная техника может переместиться за один игровой тик, без учёта типа местности и погоды. При перемещении по дуге учитывается длина дуги, а не кратчайшее расстояние между начальной и конечной точками.

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

- *getRemainingAttackCooldownTicks*
`public int` **getRemainingAttackCooldownTicks**`( )`
  - **Returns** - Возвращает количество тиков, оставшееся до следующей атаки. Для совершения атаки необходимо, чтобы это значение было равно нулю.

- *getSquaredAerialAttackRange*
`public double` **getSquaredAerialAttackRange**`( )`
  - **Returns** - Возвращает квадрат максимального расстояния (от центра до центра), на котором данная техника может атаковать воздушные объекты.

- *getSquaredGroundAttackRange*
`public double` **getSquaredGroundAttackRange**`( )`
  - **Returns** - Возвращает квадрат максимального расстояния (от центра до центра), на котором данная техника может атаковать наземные объекты.

- *getSquaredVisionRange*
`public double` **getSquaredVisionRange**`( )`
  - **Returns** - Возвращает квадрат максимального расстояния (от центра до центра), на котором данная техника обнаруживает другие объекты, без учёта типа местности и погоды.

- *getType*
`public VehicleType` **getType**`( )`
  - **Returns** - Возвращает тип техники.

- *getVisionRange*
`public double` **getVisionRange**`( )`

- **Returns** - Возвращает максимальное расстояние (от центра до центра), на котором данная техника обнаруживает другие объекты, без учёта типа местности и погоды.

- *isAerial*
`public boolean` **isAerial**`( )`
  - **Returns** - Возвращает `true` в том и только том случае, если эта техника воздушная.

- *isSelected*
`public boolean` **isSelected**`( )`
  - **Returns** - Возвращает `true` в том и только том случае, если эта техника выделена.

4.1.11 CLASS **VehicleType**

Тип техники.

DECLARATION

---

public final class VehicleType

extends Enum

FIELDS

- public static final VehicleType ARRV
  - Бронированная ремонтно-эвакуационная машина. Наземный юнит. Постепенно восстанавливает прочность находящейся поблизости неподвижной техники.
- public static final VehicleType FIGHTER
  - Истребитель. Воздушный юнит. Крайне эффективен против другой воздушной техники. Не может атаковать наземные цели.
- public static final VehicleType HELICOPTER
  - Ударный вертолёт. Воздушный юнит. Может атаковать как воздушные, так и наземные цели.
- public static final VehicleType IFV
  - Боевая машина пехоты. Наземный юнит. Может атаковать как воздушные, так и наземные цели.
- public static final VehicleType TANK
  - Танк. Наземный юнит. Крайне эффективен против другой наземной техники. Также может атаковать воздушные цели.

4.1.12 CLASS **VehicleUpdate**

Класс, частично определяющий технику. Содержит уникальный идентификатор техники, а также все поля техники, значения которых могут изменяться в процессе игры.

DECLARATION

public class VehicleUpdate

extends Object

METHODS

- *getDurability*
  `public int` **getDurability**`( )`
  - **Returns** - Возвращает текущую прочность или `0`, если техника была уничтожена либо ушла из зоны видимости.

- *getGroups*
  `public int[]` **getGroups**`( )`
  - **Returns** - Возвращает группы, в которые входит эта техника.

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

- *getRemainingAttackCooldownTicks*
  `public int` **getRemainingAttackCooldownTicks**`( )`
  - **Returns** - Возвращает количество тиков, оставшееся до следующей атаки. Для совершения атаки необходимо, чтобы это значение было равно нулю.

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

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

- *isSelected*
  `public boolean` **isSelected**`( )`
  - **Returns** - Возвращает `true` в том и только том случае, если эта техника выделена.

4.1.13 CLASS **WeatherType**

Тип погоды.

DECLARATION

public final class WeatherType

extends Enum

FIELDS

- public static final WeatherType CLEAR
  - Ясно.
- public static final WeatherType CLOUD
  - Плотные облака.
- public static final WeatherType RAIN
  - Сильный дождь.

4.1.14 CLASS **World**

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

DECLARATION

public class World

extends Object

METHODS

- *getFacilities*
  `public Facility[] getFacilities( )`

- **Returns** - Возвращает список сооружений (в случайном порядке). В зависимости от реализации, объекты, задающие сооружения, могут пересоздаваться после каждого тика.

- *getHeight*
`public double` **getHeight**`( )`

  - **Returns** - Возвращает высоту мира.

- *getMyPlayer*
`public Player` **getMyPlayer**`( )`

  - **Returns** - Возвращает вашего игрока.

- *getNewVehicles*
`public Vehicle[]` **getNewVehicles**`( )`

  - **Returns** - Возвращает список техники, о которой у стратегии не было информации в предыдущий игровой тик. В этот список попадает как только что произведённая техника, так и уже существующая, но находящаяся вне зоны видимости до этого момента.

- *getOpponentPlayer*
`public Player` **getOpponentPlayer**`( )`

  - **Returns** - Возвращает игрока, соревнующегося с вами.

- *getPlayers*
`public Player[]` **getPlayers**`( )`

  - **Returns** - Возвращает список игроков (в случайном порядке). В зависимости от реализации, объекты, задающие игроков, могут пересоздаваться после каждого тика.

- *getTerrainByCellXY*
`public TerrainType[][]` **getTerrainByCellXY**`( )`

  - **Returns** - Возвращает карту местности.

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

  - **Returns** - Возвращает базовую длительность игры в тиках. Реальная длительность может отличаться от этого значения в меньшую сторону. Эквивалентно `game.tickCount`.

- *getTickIndex*
`public int` **getTickIndex**`( )`

  - **Returns** - Возвращает номер текущего тика.

- *getVehicleUpdates*
`public VehicleUpdate[]` **getVehicleUpdates**`( )`

  - **Returns** - Возвращает значения изменяемых полей для каждой видимой техники, если хотя бы одно поле этой техники изменилось. Нулевая прочность означает, что техника была уничтожена либо ушла из зоны видимости.

- *getWeatherByCellXY*
`public WeatherType[][]` **getWeatherByCellXY**`( )`

  - **Returns** - Возвращает карту погоды.

- *getWidth*
`public double` **getWidth**`( )`

  - **Returns** - Возвращает ширину мира.

Глава 5

Package <none>

Package Contents *Page*

Interfaces

Стратегия — интерфейс, содержащий описание методов искусственного интеллекта армии.

5.1 Interfaces

5.1.1 INTERFACE Strategy

Стратегия — интерфейс, содержащий описание методов искусственного интеллекта армии. 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**`( Player` **me**`, World` **world**`, Game` **game**`, Move` **move** `)`
  - **Usage**
    - Основной метод стратегии, осуществляющий управление армией. Вызывается каждый тик.
  - **Parameters**
    - `me` - Информация о вашем игроке.
    - `world` - Текущее состояние мира.
    - `game` - Различные игровые константы.
    - `move` - Результатом работы метода является изменение полей данного объекта.