**Глава2**

## 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× *<длительность\_игры\_в\_тиках>* +10000 (2.1)

миллисекунд реального времени и

10× *<длительность\_игры\_в\_тиках>* +10000 (2.2)

миллисекунд процессорного времени.[[1]](#footnote-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*.*3 | 0*.*4 | 0*.*9 | 1*.*2 | 0*.*4 |
| 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 почти никакого влияния на игру и учитывается только при совершении некоторых действий. При включенном тумане войны стратегия участника будет получать данные только о тех юнитах противника, которые находятся в пределах дальности[[2]](#footnote-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 потенциальный урон одной атаки (*<* урон *>* − *<* защита *>*) по конкретной цели, тем с большей вероятностью эта цель будет выбрана.

Значение прочности цели не учитывается. Если потенциальный урон по цели не является положительным числом, считается, что юнит не может атаковать цель. Урон наносится мгновенно. Также считается, что все 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

*<дальность\_обзора\_юнита>* × *<коэффициент\_дальности\_обзора\_юнита>*

(2.3)

×*<коэффициент\_незаметности\_цели>*

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 8 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 3 за 60 тиков, а также уменьшает интервал между тактическими ядерными ударами на 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, не устанавливая поле move.action или инициализируя его значением NONE. Изначально количество возможных действий стратегии ограничено 12-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 скорости 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]](#footnote-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.

Как уже было упомянуто, управление в CodeWars схоже с управлением в обычных компьютерных играх жанра RTS, хотя и не претендует на полное соответствие. Стратегии доступны следующие действия:

* CLEAR\_AND\_SELECT. Стандартное выделение дружественных юнитов рамкой либо выделение ранее созданной группы юнитов. В первом случае необходимо дополнительно задать границы рамки (move.left, move.top, move.right и move.bottom), также опционально можно установить тип техники (move.vehicleType), во втором — номер группы (от 1 до 100). Если задан номер группы, то все остальные параметры объекта move будут проигнорированы.
* ADD\_TO\_SELECTION. Аналогично действию CLEAR\_AND\_SELECT с тем исключением, что существующее выделение юнитов не будет сброшено.
* DESELECT. Снимает выделение с юнитов, соответствующих указанным параметрам. Настройка действия не отличается от приведённых выше двух типов действий.
* ASSIGN. Устанавливает для всех выделенных юнитов принадлежность к группе. Необходимо дополнительно установить номер группы. При этом юниты, добавленные в группу ранее, остаются в ней. Юнит может находиться одновременно в нескольких группах.
* DISMISS. Убирает у всех выделенных юнитов принадлежность к группе. Необходимо дополнительно установить номер группы.
* DISBAND. Убирает всех юнитов из указанной группы.
* MOVE. Приказывает выделенным юнитам перемещаться в указанном направлении. Параметры move.x и move.y задают вектор движения. Таким образом, юниты перемещаются, сохраняя формацию. Дополнительно можно ограничить максимальную скорость перемещения, чтобы медленные юниты не отставали от более быстрых.
* ROTATE. Приказывает выделенным юнитам поворачиваться относительно указанной точки. Параметры move.x и move.y задают эту точку, а move.angle— угол поворота. Юниты двигаются по окружности.

Таким образом, расстояние от юнита до указанной точки не изменяется в процессе движения. Дополнительно можно ограничить максимальную линейную или максимальную угловую скорость перемещения, чтобы формация поворачивалась синхронно, а медленные юниты не отставали от более быстрых.

* SCALE. Масштабирует формацию выделенных юнитов относительно указанной точки с указанным коэффициентом. Параметры move.x и move.y задают эту точку, а move.factor— коэффициент от 0*.*1 до 10*.*0. При значениях коэффициента больше 1*.*0 происходит расширение формации, при значениях меньше 1*.*0— сжатие. Чтобы определить позицию юнита в конце движения, необходимо построить вектор из указанной точки в позицию юнита до начала движения, умножить обе координаты этого вектора на коэффициент и отложить полученный вектор от указанной в приказе точки. Все юниты будут линейно перемещаться каждый к своей цели. Дополнительно можно ограничить максимальную скорость перемещения.
* SETUP\_VEHICLE\_PRODUCTION. Настраивает производство техники на захваченном заводе. Необходимо дополнительно указать тип техники и идентификатор завода. При этом прогресс производства будет сброшен, даже если указанный тип техники равен типу техники, производящейся на заводе в данный момент.
* TACTICAL\_NUCLEAR\_STRIKE. Запрашивает нанесение тактического ядерного удара по указанным координатам. Необходимо дополнительно указать идентификатор дружественной техники, которая будет осуществлять подсветку цели. Удар можно запрашивать не чаще, чем раз в 1200 тиков. Этот интервал немного уменьшается за каждый захваченный центр управления. Нанесение удара происходит не мгновенно, а через 30 тиков после запроса. Для осуществления подсветки цель удара должна находиться в зоне видимости указанного юнита, как в момент запроса, так и все последующие тики до нанесения удара. Если юнит, подсвечивающий цель, погибает либо цель выходит из зоны видимости юнита хотя бы на один тик, нанесение удара отменяется. Удар поражает все цели на расстоянии, не превышающем 50*.*0, как вражеские, так и союзные. При этом урон в центре взрыва составляет 99 единиц и равномерно падает до нуля на краю.

## 2.7 Столкновения юнитов

Коллизия наземных юнитов между собой, а также с границами карты не допускается игровым симулятором. Коллизия воздушных 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.

**Глава3**

# Создание стратегии

## 3.1 Техническая часть

Сперва для создания стратегии вам необходимо выбрать один из ряда поддерживаемых языков программирования[[4]](#footnote-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]](#footnote-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]](#footnote-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 <Ł \_Łª Œ >, <Ł \_Łª Œ > (2), <Ł \_Łª Œ > (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 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(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() / 2.0); 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

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. [↑](#footnote-ref-1)
2. Здесь и далее под расстоянием между юнитами подразумевается расстояние между их центрами, если явно не указано другое. [↑](#footnote-ref-2)
3. Техника после перемещения частично или полностью находится за пределами карты либо пересекается с какой-либо другой техникой, такой, что столкновение этих двух юнитов запрещено правилами игры. [↑](#footnote-ref-3)
4. Для всех языков программирования используются 32-битные версии компиляторов/интерпретаторов. [↑](#footnote-ref-4)
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. [↑](#footnote-ref-5)
6. Основными параметрами формата игры являются количество игроков, участвующих в нём, и количество юнитов, находящихся под управлением каждого игрока. Кратко формат записывается в виде <Œ ºŁ æ \_Łª Œ >×<Œ ºŁ æ \_ Ł >, 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. [↑](#footnote-ref-6)