CodeTroopers 2013

Правила

Версия 1.0

Ноябрь, 2013

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

1.2 Информация об организаторе конкурса

Наименование - ООО «Мэйл.Ру»;

Адрес места нахождения: 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 04 ноября 2013 года до 24.00 часов 15 декабря 2013 года по Московскому времени.

Сроки начала и окончания этапов Конкурса:

- первый этап — с 00 часов 00 минут 23 ноября 2013 года до 24 часов 00 минут 24 ноября 2013 года;
- второй этап — с 00 часов 00 минут 30 ноября 2013 года до 24 часов 00 минут 1 декабря 2013 года;
- третий этап (заключительный) — с 00 часов 00 минут 7 декабря 2013 года до 24 часов 00 минут 8 декабря 2013 года;

1.4 Условие получения статуса Участника конкурса

Для участия в Конкурсе необходимо пройти процедуру регистрации в Системе Организатора конкурса размещенной 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 4 ноября 2013 года до 24.00 часов 15 декабря 2013 года включительно.

1.6 Территория проведения Конкурса

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

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

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

Конкурсная документация включает в себя:

- Объявление о проведении Конкурса;
- Соглашение об организации и порядке проведения Конкурса;
- Правила проведения Конкурса;
- информационные данные, содержащиеся в Системе Организатора конкурса.

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

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

1.8 Порядок определения Победителей и вручения Призов. Призовой фонд Конкурса

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

Призовой фонд Конкурса формируется за счет средств Организатора конкурса.

Призовой фонд:

- 1-е место: Apple Macbook Pro 15"Retina
- 2-3 места: Apple Macbook Air 13"
- 4-6 места: Apple iPad mini 64Gb Wi-Fi + Cellular
- 1-6 места в квалификации (Песочница): Apple iPod Touch 32 GB

Все участники Конкурса, принявшие участие во втором этапе, будут вознаграждены футболкой.

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, в течение двух недель после окончания финального этапа. Срок доставки приза по почтовому адресу, указанному участником, зависит от сроков доставки Почты России. Почтовые адреса призеров, для отправки призов, Организатор получает из учетных данных участника в Системе Организатора. Адрес должен быть указан участником-призером в течение трех 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 выдается

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

- копию паспорта Победителя;
- копию свидетельства о постановки 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 отчетности о проведенном 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 несет все расходы по уплате всех применимых налогов, установленных действующим законодательством Российской Федерации.

1.9 Порядок и способ информирования участников Конкурса

Информирование Участников Конкурса осуществляется путем 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 CodeTroopers 2013

2.1 Общие положения и правила

Данное соревнование предоставляет вам возможность проверить свои навыки программирования, создав искусственный интеллект (стратегию), управляющий небольшим отрядом бойцов в специальном игровом мире (подробнее об особенностях мира CodeTroopers 2013 можно узнать в следующих разделах). Вы должны сражаться против одного или нескольких других игроков в режиме «каждый сам за себя (КСЗС)». У игрока в распоряжении находится отряд из нескольких (от 3 до 5 в зависимости от этапа соревнования) бойцов (юнитов[1]), обладающих разными характеристиками и способностями. Наборы юнитов являются одинаковыми для каждого из игроков. Юниты получают право ходить по очереди. Ход следующего юнита начинается только после завершения хода предыдущего. Очерёдность ходов определяется составленным в начале и неизменным по ходу игры списком бойцов (убитые бойцы пропускаются). Для составления этого списка все игроки перемешиваются случайным образом, затем берётся случайный тип бойца из присутствующих на поле, и в список очерёдности добавляется по одному бойцу данного типа для каждого игрока согласно порядку игроков, полученному в результате перемешивания. Затем берётся новый случайный тип бойца (из оставшихся), и операция повторяется до тех пор, пока в списке не окажутся все бойцы. Игроки перемешиваются только один раз, таким образом никакие два юнита одного игрока не могут совершать ход подряд при условии, что ни один боец на поле не был убит. Игра завершается по истечении определённого количества ходов или в том случае, когда на поле остались юниты только одного игрока. Места между игроками распределяются в соответствии с количеством набранных баллов. Подробнее система начисления баллов описана далее.

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 боях.

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

[1] Строго говоря, бойцы являются подмножеством юнитов. В игре есть ещё один класс юнитов: бонусы. Подробнее о бонусах читайте далее.

эффективнее.

Начальное значение рейтинга на каждом основном этапе турнира равно 0. За каждый бой участник получает определённое количество единиц рейтинга в зависимости от занятого в бою места (система, аналогичная используемой в чемпионате Формула-1). Если двое или более участников делят какое-то место, то суммарное количество единиц рейтинга за это место и за следующие `количество_таких_участников` − 1 мест делится поровну между этими участниками. Например, если двое участников делят третье место, то каждый из них получит половину от суммы единиц рейтинга за третье и четвёртое места. При делении округление всегда совершается в меньшую сторону. Более подробная информация об этапах турнира будет представлена в анонсах на сайте проекта.

Сначала все участники могут участвовать только в боях, проходящих в Песочнице. Игроки могут отправлять в Песочницу свои стратегии, и последняя принятая из них берётся системой для участия в квалификационных боях. Каждый игрок участвует примерно в 1 квалификационном бою за час. Жюри оставляет за собой право изменить этот интервал, исходя из пропускной способности тестирующей системы, однако для всех игроков он остаётся постоянной величиной. Бои в Песочнице проходят в формате[2], соответствующем формату случайного прошедшего этапа турнира или же формату следующего (текущего) этапа. При этом, чем ближе значение рейтинга двух игроков в рамках Песочницы, тем больше вероятность того, что они окажутся в одном бою. 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 боёв в Песочнице происходит отбор для участия в Раунде 1, в который пройдут 900 участников с наибольшим рейтингом (при его равенстве приоритет отдаётся игроку, раньше отправившему последнюю версию своей стратегии).

Этапы турнира:

- Раунд 1 проверит ваши навыки управления отрядом из трёх бойцов, который будет помещён на игровое поле вместе с отрядами трёх других игроков (4×3). Координировать действия небольшого отряда не так сложно, однако потеря каждого бойца серьёзно подрывает боеспособность отряда. Испытание вашей стратегии начинается уже здесь. Этот этап, как и все последующие, состоит из двух частей, между которыми будет небольшой перерыв (с возобновлением работы Песочницы), который позволит улучшить свою стратегию. Для боёв в каждой части выбирается последняя стратегия, отправленная игроком до начала части. Бои проводятся волнами. В каждой волне каждый игрок участвует ровно в одном бою. Количество волн в каждой части определяется возможностями тестирующей системы, но гарантируется, что оно не будет меньше 10. 300 участников с наиболее высоким рейтингом пройдут в Раунд 2. Также в Раунд 2 будет проведен добор 60 участников с наибольшим рейтингом в Песочнице из числа тех, кто не прошёл по итогам Раунда 1.
- В Раунде 2 в каждом бою будет участвовать по 4 игрока, выставляющих увеличенные отряды из 4 бойцов (тип боёв 4×4). Между этапами будет некоторый перерыв, так что у вас будет возможность немного доработать стратегию, улучшить координацию между бойцами и учесть специальные способности нового юнита. Усложняет задачу то, что после подведения итогов Раунда 1 часть слабых стратегий будет отсеяна и вам придётся противостоять более сильным соперникам. По итогам Раунда 2 лучшие 50 стратегий попадут в Финал. Также в Финал будет проведен добор 10 участников с наибольшим рейтингом в Песочнице из числа тех, кто не прошёл в рамках основного турнира.
- Финал является самым серьёзным этапом. После отбора, проведённого по итогам двух первых этапов, останутся сильнейшие. И в каждом бою вам придётся сойтись лицом к лицу с одним из них. У игрока в распоряжении находится отряд из 5 бойцов (тип боёв 2×5), и здесь взаимодействие бойца с товарищами может оказаться решающим фактором. Одна ошибка — и вся ваша команда будет уничтожена. 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 дуэли между всеми парами участников Финала. Если позволит время и возможности тестирующей системы, операция будет повторена.

[2]Под форматом боя понимается совокупность таких параметров, как количество игроков, участвующих в нём, и количество бойцов в команде каждого игрока. Кратко формат записывается в виде `<количество_игроков>` × `<количество_бойцов>`, например запись 4×3 означает формат боя, в котором участвует 4 игрока, управляющих тремя бойцами каждый.

После окончания Финала все финалисты упорядочиваются по невозрастанию рейтинга. При равенстве рейтингов более высокое место занимает тот финалист, чья участвовавшая в финале стратегия была отослана раньше. Призы за Финал распределяются на основании занятого места после этого упорядочивания. Лучшие шесть финалистов награждаются призами:

- 1 место — Apple Macbook Pro 15"Retina,
- 2-3 места — Apple Macbook Air 13",
- 4-6 места — Apple iPad mini 64Gb Wi-Fi + Cellular.

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

В игре существует 5 типов бойцов, различающихся по целому ряду параметров: командир, полевой медик, штурмовик, снайпер, разведчик. Однако игрок не может их выбирать. В зависимости от этапа соревнования, ему будет предоставлен фиксированный набор бойцов. Каждый тип бойца будет представлен не более, чем одним юнитом.

2.2 Описание игрового мира

Поле игры представляет собой прямоугольник, разделённый на квадратные клетки. Количество клеток 30×20. В игре используется система координат, ось абсцисс в которой направлена слева направо, а ось ординат — сверху вниз. Верхняя левая клетка имеет координаты $(0, 0)$. Существует набор подготовленных вариантов поля, отличающихся различными параметрами, такими, как начальное положение юнитов и уровень высоты клеток. Для каждой игры поле выбирается случайно из набора. В процессе проведения чемпионата могут быть добавлены новые карты, а также внесены изменения в уже существующие. Гарантируется, что каждая карта симметрична относительно горизонтальной и вертикальной осей, проходящих через её центр и делящих таким образом карту на 4 равные части. Боевые подразделения игроков расположены симметрично каждое в своей четверти карты[3]. Координаты клеток начального расположения юнитов зависят от конкретной карты и для этой карты не меняются от игры к игре. Юнит занимает ровно одну клетку. В игре есть два крупных класса юнитов: бойцы и бонусы. Два юнита одного класса не могут одновременно находиться в одной клетке.

Клетка поля может иметь один из четырёх уровней высоты:

- Уровень 0: не содержит препятствие (free). Клетка проходима для юнитов и не предоставляет никакой защиты от выстрелов.
- Уровень 1: содержит низкое препятствие (low cover). Клетка данного, как и всех последующих, типов является непроходимой для юнитов и её можно использовать для защиты от выстрелов, о чём подробнее читайте далее.
- Уровень 2: содержит препятствие средней высоты (medium cover).
- Уровень 3: содержит высокое препятствие (high cover).

Как уже упоминалось выше, игра является пошаговой. Все бойцы получают право хода по очереди, согласно определённому в начале игры и далее неизменному порядку.

Длительность игры измеряется в игровых ходах — *больших итерациях*, за каждую из которых все бойцы один раз совершают свой ход. Максимальная длительность каждой игры — 50 таких ходов. Существуют также случаи, в которых игра может закончиться раньше этого времени, например, в случае, если все живые

[3] Если в игре принимают участие два игрока, их отряды занимают левую верхнюю и правую нижнюю четверти карты.

бойцы на поле принадлежат одному игроку, либо стратегии всех игроков «упали». «Упавшая» стратегия больше не может управлять бойцами.

Ход каждого бойца является *малой итерацией*, состоящей из одного или нескольких действий, которые боец совершает до тех пор, пока у него достаточно очков действия или он не завершит ход принудительно. Стратегия, управляющая бойцом, получает слепок игрового мира и отвечает на него указанием, какое действие боец должен совершить. Игровой симулятор моделирует это действие, если оно не противоречит каким-либо ограничениям. Затем стратегия получает обновлённый слепок мира, отвечает на него и так далее до конца хода.

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 попыталась совершить недопустимое действие[4].
- Стратегия превысила одно (любое) из отведённых ей ограничений по времени. Стратегии на один ход бойца выделяется не более 2 секунд реального времени. Но в сумме на всю игру процессу стратегии выделяется $500 \times <длительность\_игры\_в\_ходах> \times <количество\_бойцов\_в\_отряде>$ миллисекунд реального времени и $200 \times <длительность\_игры\_в\_ходах> \times <количество\_бойцов\_в\_отряде>$ миллисекунд процессорного времени.[5] 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.

В мире CodeTroopers 2013 существует 2 класса юнитов:

- Боец — юнит, управляемый игроком (подробнее управление бойцом будет описано далее).
- Бонус (полезный предмет) — статический объект. Боец автоматически подбирает бонус, если находится с ним в одной клетке после совершения своего действия и уже не несёт с собой бонус данного типа. Все бонусы случайным образом помещаются на карту до начала игры. Гарантируется, что количество и расположение бонусов симметрично для каждой четверти карты.

2.3 Описание бойцов

Сперва определим понятие *досягаемости* юнита Б для юнита А: оно будет использовано далее. Юнит Б является досягаемым для юнита А, если из центра клетки, в которой находится юнит А, можно провести отрезок хотя бы в одну точку клетки юнита Б, такой, что он не проходит ни через одну клетку (или даже её угол) с высотой препятствия, равной или превосходящей высоту хотя бы одного из этих двух юнитов. Из определения следует, что досягаемость юнита Б для юнита А не означает досягаемости юнита А для юнита Б[6].

Высота бойца определяется согласно его положению/стойке:

- Положение лёжа (prone). В этом положении высота бойца соответствует высоте низкого препятствия.
- Положение сидя (kneeling). Высота бойца соответствует высоте среднего препятствия.
- Положение стоя (standing). Высота бойца соответствует высоте высокого препятствия.

Каждый боец обладает следующим набором характеристик:

[4] Подробнее о случаях, приводящих к «падению» стратегии, можно узнать из документации к действиям.

[5] 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.

[6] Участникам не нужно реализовывать собственный алгоритм проверки на досягаемость, для этого есть метод «`isVisible`» объекта «`World world`» (стиль именования Java).

- Очки здоровья (hitpoints). Если очки здоровья бойца падают до нуля, то он считается погибшим и убирается с поля. Начальное количество очков здоровья зависит только от типа бойца (читайте о типах бойцов далее).
- Очки действия (action points). Очки действия — это ресурс, который боец тратит на смену стойки, перемещение по карте, стрельбу и использование способностей, специфических для конкретного типа бойцов. Количество очков действия бойца в начале его хода является константным для типа бойца. Для смены положения на один уровень боец тратит 2 очка действия. Таким образом для перевода бойца из состояния лёжа в состояние стоя необходимо 4 очка действия. Для перемещения на одну клетку по вертикали или горизонтали необходимо 2 очка действия в положении стоя, 4 — в положении сидя и 6 — в положении лёжа. Перемещение по диагонали в игре не предусмотрено.
- Дальность обзора (vision range). Максимальная дистанция[7], на которой боец может обнаружить противника. Противник считается видимым для бойца, если он находится в пределах дальности обзора и является досягаемым для бойца. Видимость противника для одного бойца игрока означает также видимость для всех юнитов игрока. Таким образом бойцы с бóльшим обзором могут «подсвечивать» противников для бойцов с бóльшей дальностью стрельбы, но меньшим обзором.
- Дальность стрельбы (shooting range). Максимальная дистанция, на которой боец может выстрелить в противника. Противник также должен быть досягаемым для юнита. Боец может выбрать для стрельбы любую клетку, находящуюся в радиусе стрельбы. Если в ней находится противник и противник является досягаемым, то ему нанесётся урон. Противник не обязательно должен быть видимым для юнитов игрока. Пересечение линией стрельбы клеток с другими юнитами игнорируется.
- Урон от выстрела (damage). Количество очков здоровья, которое противник теряет после выстрела бойца в него. Урон зависит от положения стрелка.
- Стоимость выстрела (shot cost). Количество очков действия, необходимое бойцу для совершения одного выстрела.
- Начальный бонус. Тип бонуса, который выдаётся бойцу в начале игры. Более подробно о бонусах можно узнать в следующем разделе правил.
- Особая способность бойца. Особая характеристика или действие, которое может совершить юнит. Зависит от типа бойца.

Как уже было сказано выше, в игре существует пять типов бойцов. В следующей таблице приведены их основные характеристики. В графе «Очки здоровья» левое число означает количество очков здоровья, которое боец имеет в начале игры, а правое — максимальное количество очков здоровья, до которого можно вылечить бойца.

Характеристика бойца	Командир	Полевой медик	Штурмовик	Снайпер	Разведчик
Очки здоровья (нач./макс.)	100/100	100/100	120/100	100/100	100/100
Очки действия	10	10	10	10	12
Дальность обзора	8	7	7	7	9
Дальность стрельбы	7	5	8	10	6
Урон от выстрела (стоя)	15	9	25	65	20
Урон от выстрела (сидя)	20	12	30	80	25
Урон от выстрела (лёжа)	25	15	35	95	30
Стоимость выстрела	3	2	4	9	4
Начальный бонус	—	Аптечка	Граната	—	Сухой паёк

Помимо основных характеристик у каждого типа бойца есть умение или свойство, которое качественно отличает его от других.

Одним своим присутствием командир (commander) способен поднять боевой дух солдат. Все юниты игрока в радиусе 5 клеток от командира получают в начале своего хода на 2 очка действия больше, чем обычно. Не распространяется на самого командира и разведчика. Также, потратив 10 очков действия, командир может отправить запрос в штаб на вылет самолёта-разведчика. В результате этого, в начале следующего своего хода командир получит примерные данные о расположении противника. Для каждого игрока будет вычислено среднее арифметическое значение координат X и Y всех его живых бойцов (при делении округление производится в меньшую сторону). Затем к каждому из полученных значений добавится

[7] Здесь и далее под дистанцией/расстоянием подразумевается евклидово расстояние между центрами клеток.

случайное отклонение, по модулю не превышающее 5, и результат запишется в специальные поля каждого игрока, о чём подробнее можно узнать в главе «Package model». Командир входит в отряд игрока на всех этапах соревнования.

Полевой медик (field medic) способен залечивать раны бойцов прямо на поле боя. Потратив одно очко действия, он может восполнить 5 единиц здоровья юниту, находящемуся на выбранной соседней (по вертикали или горизонтали) клетке, или 3 единицы здоровья себе. Здоровье юнита не может стать больше 100. Полевой медик входит в отряд игрока на всех этапах соревнования.

Для прорыва хорошо укреплённой линии обороны противника штурмовик (soldier) оснащён стальным нагрудником, который способен принять на себя некоторое количество урона — начальное количество очков здоровья у штурмовика на 20 больше, чем у любого другого юнита. Однако штурмовик может быть излечен полевым медиком или аптечкой первой помощи только до уровня в 100 очков здоровья. Штурмовик входит в отряд игрока на всех этапах соревнования.

Снайпер (sniper) в совершенстве владеет искусством маскировки. В процессе проверки видимости снайпера, находящегося в положении сидя, эффективный радиус обзора юнитов противника считается меньше табличного на 0.5 клетки. Если снайпер находится в положении лёжа, то штраф составляет 1 клетку. Также в положении сидя дальность стрельбы снайпера увеличивается на 1, а в положении лёжа — на 2. Снайпер входит в отряд игрока, начиная с этапа турнира Раунд 2.

Намётанный глаз разведчика (scout) позволяет отличать замаскированные цели: разведчик не получает штрафа к дальности обзора при обнаружении снайперов противника. У разведчика на 2 очка действия больше, чем у других бойцов, однако он привык действовать самостоятельно и потому не получает дополнительных очков действия рядом с командиром. Разведчик входит в отряд игрока, начиная с этапа турнира Финал.

2.4 Описание бонусов

Бонусы - особые предметы, которые бойцы могут использовать для получения преимуществ в бою. Как уже было сказано выше, бонусы появляются на карте в начале игры и находятся на своих местах до её конца или до того, как их кто-то подберёт. При проверке видимости бонуса его высота считается равной высоте бойца в положении лёжа.

Типы бонусов:

- Граната (grenade). Потратив 8 очков действия, боец может бросить гранату в любую клетку, расстояние до центра которой от центра его клетки не превышает 5. Цель не обязательно должна быть досягаема для бойца. Юнит в целевой клетке получает 80 очков урона, а все юниты в соседних (по вертикали или горизонтали) клетках — по 60.
- Аптечка (medikit). Потратив 2 очка действия, боец может восполнить 50 единиц здоровья юниту, находящемуся на выбранной соседней (по вертикали или горизонтали) клетке, или 30 единиц здоровья себе. Здоровье юнита не может стать больше 100.
- Сухой паёк (field ration). Потратив 2 очка действия, боец может восполнить себе 5 очков действия. Таким образом, суммарная прибавка равна 3. Количество очков действия не может стать больше начального (без учёта командирского бонуса[8]) для данного типа юнита.

Юнит может нести одновременно не более одного бонуса каждого типа.

Получить бонусы можно одним из 2 способов:

- Некоторые типы бойцов в начале игры автоматически получают бонус определённого типа.
- Боец, оказавшийся в клетке с бонусом, автоматически подбирает его, если у него ещё нет бонуса данного типа. Иначе бонус остаётся на месте.

[8]Например, для данного типа бойца начальное количество очков действия составляет 10, сейчас начало его хода, и он находится рядом с командиром. Тогда количество очков действия у него 12. Если он съест сухой паёк, то потратит на это 2 очка действия, однако никакого бонуса он не получит, так как количество его очков действия уже равно начальному.

2.5 Начисление баллов

Вернёмся к теме начисления баллов игрокам. Количество набранных игроками баллов — единственный фактор, который определяет победителя (или занятое место). На рейтинг игрока в рамках турнира влияет только занятое им в бою место, но не количество баллов.

Игрок получает баллы за следующие действия:

1. Юнит игрока нанёс урон юниту противника. За каждую единицу урона игрок получает 1 балл.
2. Юнит игрока уничтожил юнит противника. 25 баллов. Баллы даются за сам факт уничтожения. За нанесённый урон игрок получает баллы согласно пункту 1.
3. Юнит игрока уничтожил последнего юнита другого игрока. 50 баллов. Действие пункта 2 игнорируется.
4. Юнит игрока уничтожил последнего противника на поле боя или игрок остался последним выжившим (например, в случае «самоподрыва» противника). 100 баллов. Действие пунктов 2 и 3 игнорируется.

Глава 3

Создание стратегии

3.1 Техническая часть

Сперва для создания стратегии вам необходимо выбрать один из ряда поддерживаемых языков программирования: Java (Oracle JDK 7), C# (Mono 2.10+), C++ (GNU MinGW C++ 4.4), Python 2 (Python 2.7+), Python 3 (Python 3.3+), Pascal (Free Pascal 2.6). 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)[9]. 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 окно.

После того, как вы отправили свою стратегию, она попадает в очередь тестирования. Система сперва попытается скомпилировать пакет с вашими файлами, а затем, если первая операция прошла успешно, создать несколько коротких (по 3 хода) игр разных форматов: 4×3, 4×4 и 2×5. Для управления каждым отрядом бойцов 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 сражениях будут даны имена в формате `<имя_игрока>`, `<имя_игрока> (2)`, `<имя_игрока> (3)` и т.д.

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

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

- Нельзя отправлять стратегию чаще, чем три раза за пять минут.
- За пять минут нельзя создать более двух пользовательских игр.

Для упрощения отладки небольших изменений стратегии в системе присутствует возможность сделать

[9] 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.

тестовую посылку (флажок «Тестовая посылка» на форме отправки стратегии). Тестовая посылка не отображается другим пользователям, не участвует в квалификационных боях в Песочнице и боях в этапах турнира, также невозможно собственноручно создавать бои с её участием. Однако после принятия данной посылки система автоматически добавляет тестовую игру с четырьмя участниками (формат 4×3): непосредственно тестовой посылкой, стратегией из раздела «Быстрый старт» и двумя стратегиями, которые ничего не делают. Тестовая игра видна только участнику, сделавшему данную тестовую посылку. Максимальная длительность такой тестовой игры составляет 25 ходов. 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 бои. Для этого нужно нажать кнопку «Смотреть» в списке боёв либо нажать кнопку «Посмотреть игру» на странице боя.

Если вы смотрите бой 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 данного боя. 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 логгирование и наблюдать за реакцией вашей стратегии в каждый момент игры. Для этого скачайте Repeater с сайта CodeTroopers 2013 (раздел «Документация» → «Утилита Repeater») и разархивируйте. Для запуска Repeater вам необходимо установленное ПО Java 7 Runtime Environment. Обратите внимание, что любое взаимодействие вашей стратегии с игровым миром при локальном повторе полностью игнорируется. Это означает, что в каждый момент времени окружающий мир для стратегии в точности совпадает с миром, каким он был в игре при тестировании на сервере и не зависит от того, какие действия ваша стратегия предпринимает. Подробнее об утилите Repeater читайте в соответствующем разделе на сайте.

Помимо всего выше перечисленного у игроков есть возможность запускать простые тестовые игры локально на своём компьютере. Для этого необходимо загрузить архив с утилитой Local runner из раздела сайта «Документация» → «Local runner». Использование данной утилиты позволит вам тестировать свою стратегию в условиях, аналогичных условиям тестовой игры на сайте, но без каких либо ограничений по количеству создаваемых игр. Длительность подобных локальных игр составляет стандартные 50 ходов. Рендерер для локальных игр заметно отличается от рендерера на сайте. Все игровые объекты в нём отображаются схематично (без использования красочных моделей). Создать локальную тестовую игру очень просто: запустите Local runner с помощью соответствующего скрипта запуска (для Windows или *n*x систем), затем запустите свою стратегию из среды разработки (или любым другим удобным вам способом) и смотрите бой. 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 10 минут. По прошествии этого времени он посчитает стратегию «упавшей» и продолжит работу без неё.

3.2 Управление бойцом

Стратегия управляет бойцом, определяя, какое действие он должен совершить. Действие может быть ненаправленным, например, подняться/опуститься (raise stance/lower stance), тогда этого достаточно; либо действие может быть направленным, например, переместиться (move) или выстрелить (shoot) и в этом случае нужно указать направление (direction) или точные координаты (x и y) клетки, на которую направлено действие. С помощью установки направления можно конкретизировать только действия, направленные на собственную клетку юнита или соседнюю с ней. Это удобно, если мы хотим переместить бойца. Для стрельбы же нужно задать точные координаты цели, хотя для стрельбы в соседнюю клетку допустимо также задавать направление. Игровой симулятор в первую очередь пытается использовать направление. Если оно не задано, то будут использованы точные координаты.

Управление бойцом осуществляется с помощью метода «`move`»[10] стратегии, который вызывается для

[10] Здесь и далее используется стиль именования Java, если только явно не указано, о каком языке программирования идёт речь.

совершающего ход бойца несколько раз до тех пор, пока не будут истрачены все очки действия или ход не будет завершён принудительно (end turn). В качестве параметров методу передаются боец «`Trooper self`», для которого вызывается метод, текущее состояние мира «`World world`», набор игровых констант «`Game game`» и объект «`Move move`», устанавливая свойства которого, стратегия и определяет поведение бойца.

3.3 Примеры реализации

Далее для всех языков программирования приведены простейшие примеры стратегий, которые перемещают бойца в случайном направлении до тех пор, пока у него не закончатся очки действия. Полную документацию классов и методов для языка Java можно найти в следующих главах.

3.3.1 Пример для Java

```
import model.*;

import java.util.Random;

public final class MyStrategy implements Strategy {
    private final Random random = new Random();

    @Override
    public void move(Trooper self, World world, Game game, Move move) {
        if (self.getActionPoints() < game.getStandingMoveCost()) {
            return;
        }

        move.setAction(ActionType.MOVE);

        if (random.nextBoolean()) {
            move.setDirection(random.nextBoolean() ? Direction.NORTH : Direction.SOUTH);
        } else {
            move.setDirection(random.nextBoolean() ? Direction.WEST : Direction.EAST);
        }
    }
}
```

3.3.2 Пример для C#

```
using System;
using Com.CodeGame.CodeTroopers2013.DevKit.CSharpCgdk.Model;

namespace Com.CodeGame.CodeTroopers2013.DevKit.CSharpCgdk
{
    public sealed class MyStrategy : IStrategy
    {
        private readonly Random random = new Random();

        public void Move(Trooper self, World world, Game game, Move move)
        {
            if (self.ActionPoints < game.StandingMoveCost)
            {
                return;
            }

            move.Action = ActionType.Move;

            if (random.Next(2) == 0)
            {
                move.Direction = random.Next(2) == 0 ? Direction.North : Direction.South;
            }
            else
            {
                move.Direction = random.Next(2) == 0 ? Direction.West : Direction.East;
            }
        }
    }
}
```

3.3.3 Пример для C++

```
#include "MyStrategy.h"

#define _USE_MATH_DEFINES
#include <cmath>
#include <cstdlib>

using namespace model;
using namespace std;

MyStrategy::MyStrategy() { }

void MyStrategy::move(const Trooper& self, const World& world, const Game& game, Move& move) {
    if (self.getActionPoints() < game.getStandingMoveCost()) {
        return;
    }

    move.setAction(MOVE);

    if (rand() % 2 == 0) {
        move.setDirection(rand() % 2 == 0 ? NORTH : SOUTH);
    } else {
        move.setDirection(rand() % 2 == 0 ? WEST : EAST);
    }
}
```

3.3.4 Пример для Python 2 и Python 3

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

```
from random import getrandbits
from model.ActionType import ActionType
from model.Direction import Direction
from model.Game import Game
from model.Move import Move
from model.Trooper import Trooper
from model.World import World


class MyStrategy:
    def move(self, me, world, game, move):
        if me.action_points < game.standing_move_cost:
            return

        move.action = ActionType.MOVE

        if getrandbits(1):
            move.direction = Direction.NORTH if getrandbits(1) else Direction.SOUTH
        else:
            move.direction = Direction.WEST if getrandbits(1) else Direction.EAST
```

3.3.5 Пример для Pascal

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

```
unit MyStrategy;

interface

uses
    StrategyControl, TrooperControl, WorldControl, MoveControl, GameControl;

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

    end;

implementation

uses
    Math;

procedure TMyStrategy.Move(me: TTrooper; world: TWorld; game: TGame; var move: TMove);
begin
    if (me.GetActionPoints() < game.GetStandingMoveCost()) then begin
        Exit;
    end;

    move.SetAction(MOVE_ACTION);

    if (Random(2) <> 0) then begin
        if Random(2) <> 0 then begin
            move.SetDirection(NORTH);
        end else begin
            move.SetDirection(SOUTH);
        end;
    end else begin
        if Random(2) <> 0 then begin
            move.SetDirection(WEST);
        end else begin
            move.SetDirection(EAST);
        end;
    end;
end;

end.
```

Глава 4

Package model

Package Contents *Page*

4.1 Classes

4.1.1 CLASS **ActionType**

Возможные действия бойца.

DECLARATION

public final class ActionType

extends Enum

FIELDS

- public static final ActionType END_TURN
  - Передать ход следующему бойцу.
- public static final ActionType MOVE
  - Переместить бойца в указанную соседнюю (по вертикали или горизонтали) клетку.

    Клетку можно указать, установив направление к ней (`Move.direction`) или её координаты (`Move.x` и `Move.y`). Для симулятора игры направление является более приоритетным.

    Если для перемещения не хватает очков действия, то стратегия игрока считается упавшей. Если боец не может переместиться (достигнута граница игрового поля или указанная клетка занята), то боец не меняет свою позицию, однако очки действия всё равно тратятся.
- public static final ActionType SHOOT
  - Выстрелить в указанную клетку.

    Клетку можно указать, установив направление к ней (`Move.direction`), если клетка является соседней, или её координаты (`Move.x` и `Move.y`). Для симулятора игры направление является более приоритетным.

    Если для выстрела не хватает очков действия, то стратегия игрока считается упавшей. Если указанная клетка находится за пределами игрового поля, находится за пределом дальности стрельбы данного бойца или же является клеткой расположения данного бойца, то стратегия игрока считается упавшей.
- public static final ActionType RAISE_STANCE
  - Сменить стойку бойца на более высокую.

    Из положения лёжа (`TrooperStance.PRONE`) боец перейдёт в положение сидя/пригнувшись (`TrooperStance.KNEELING`). Из положения сидя/пригнувшись (`TrooperStance.KNEELING`) боец перейдёт в положение стоя (`TrooperStance.STANDING`).

    Если для смены стойки не хватает очков действия, то стратегия игрока считается упавшей. Если боец находится в положении стоя (`TrooperStance.STANDING`), то его стойка не меняется, однако очки действия всё равно тратятся.
- public static final ActionType LOWER_STANCE

- Сменить стойку бойца на более низкую.

  Из положения стоя (`TrooperStance.STANDING`) боец перейдёт в положение сидя/пригнувшись (`TrooperStance.KNEELING`). Из положения сидя/пригнувшись (`TrooperStance.KNEELING`) боец перейдёт в положение лёжа (`TrooperStance.PRONE`).

  Если для смены стойки не хватает очков действия, то стратегия игрока считается упавшей. Если боец находится в положении лёжа (`TrooperStance.PRONE`), то его стойка не меняется, однако очки действия всё равно тратятся.

- public static final ActionType THROW_GRENADE
  - Бросить гранату в указанную клетку. Граната изымается у бойца, совершающего ход.

    Клетку можно указать, установив направление к ней (`Move.direction`), если клетка является соседней, или её координаты (`Move.x` и `Move.y`). Для симулятора игры направление является более приоритетным.

    Если для броска гранаты не хватает очков действия, то стратегия игрока считается упавшей. Если боец не имеет гранаты, то стратегия игрока считается упавшей. Если указанная клетка находится за пределами игрового поля или находится за пределом дальности броска гранаты, то стратегия игрока считается упавшей.

- public static final ActionType USE_MEDIKIT
  - Вылечить себя или бойца в указанной соседней (по вертикали или горизонтали) клетке с помощью аптечки первой помощи. Аптечка изымается у бойца, совершающего ход.

    Клетку можно указать, установив направление к ней (`Move.direction`), если клетка является соседней, или её координаты (`Move.x` и `Move.y`). Для симулятора игры направление является более приоритетным.

    Если для использования аптечки не хватает очков действия, то стратегия игрока считается упавшей. Если боец не имеет аптечки, то стратегия игрока считается упавшей. Если указанная клетка находится за пределами игрового поля или не является собственной клеткой бойца или соседней с ней, то стратегия игрока считается упавшей. Если в указанной клетке нет бойца, то боец, совершающий ход, сохраняет аптечку, однако очки действия всё равно тратятся.

- public static final ActionType EAT_FIELD_RATION
  - Съесть сухпаёк. Сухпаёк изымается у бойца, совершающего ход.

    Если для использования сухпайка не хватает очков действия, то стратегия игрока считается упавшей. Если боец не имеет сухпайка, то стратегия игрока считается упавшей.

- public static final ActionType HEAL
  - Умение полевого медика: вылечить себя или бойца в указанной соседней (по вертикали или горизонтали) клетке.

    Клетку можно указать, установив направление к ней (`Move.direction`), если клетка является соседней, или её координаты (`Move.x` и `Move.y`). Для симулятора игры направление является более приоритетным.

    Если для использования умения не хватает очков действия, то стратегия игрока считается упавшей. Если боец, совершающий ход, не является полевым медиком, то стратегия игрока считается упавшей. Если указанная клетка находится за пределами игрового поля или не является собственной клеткой медика или соседней с ней, то стратегия игрока считается упавшей. Если в указанной клетке нет бойца, то медик ничего не делает, однако очки действия всё равно тратятся.

- public static final ActionType REQUEST_ENEMY_DISPOSITION
  - Умение командира: отправить в штаб запрос на вылет самолёта-разведчика. В результате, в начале следующего хода командира в полях `Player.approximateX` и `Player.approximateY` каждого игрока будут содержаться координаты примерного расположения его бойцов. Значение каждой координаты будет равно среднему арифметическому значению соответствующих координат юнитов игрока на поле со случайным смещением, не превышающим по модулю 5. Гарантируется, что полученная точка лежит в пределах игрового поля.

Если для использования умения не хвататет очков действия, то стратегия игрока считается упавшей. Если боец, совершающий ход, не является командиром, то стратегия игрока считается упавшей.

METHODS

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

4.1.2 CLASS **Bonus**

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

DECLARATION

public final class Bonus

extends Unit

METHODS

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

4.1.3 CLASS **BonusType**

Тип бонуса.

DECLARATION

public final class BonusType

extends Enum

FIELDS

- public static final BonusType GRENADE
  - Граната. Потратив 8 очков действия, боец может бросить гранату в любую клетку, расстояние до центра которой от центра его клетки не превышает 5. Цель не обязательно должна быть досягаема для бойца. Юнит в целевой клетке получает 80 очков урона, а все юниты в соседних (по вертикали или горизонтали) клетках — по 60.
- public static final BonusType MEDIKIT
  - Аптечка. Потратив 2 очка действия, боец может восполнить 50 единиц здоровья юниту, находящемуся на выбранной соседней (по вертикали или горизонтали) клетке, или 30 единиц здоровья себе. Здоровье юнита не может стать больше 100.
- public static final BonusType FIELD_RATION
  - Сухой паёк. Потратив 2 очка действия, боец может восполнить себе 5 очков действия. Таким образом суммарная прибавка равна 3. Количество очков действия не может стать больше начального (без учёта командирского бонуса) для данного типа юнита.

METHODS

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

4.1.4 CLASS **CellType**

Тип клетки игрового поля.

DECLARATION

public final class CellType **extends** Enum

FIELDS

- public static final CellType FREE
  - Клетка проходима для юнитов.

- public static final CellType LOW_COVER
  - Клетка содержит низкое препятствие и не проходима для юнитов.
- public static final CellType MEDIUM_COVER
  - Клетка содержит препятствие средней высоты и не проходима для юнитов.
- public static final CellType HIGH_COVER
  - Клетка содержит высокое препятствие и не проходима для юнитов.

METHODS

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

4.1.5 CLASS **Direction**

Направление.

DECLARATION

public final class Direction **extends** Enum

FIELDS

- public static final Direction CURRENT_POINT
  - Текущая клетка.
- public static final Direction NORTH
  - Клетка к северу.
- public static final Direction EAST
  - Клетка к востоку.
- public static final Direction SOUTH
  - Клетка к югу.
- public static final Direction WEST
  - Клетка к западу.

METHODS

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

4.1.6 CLASS Game

Предоставляет доступ к различным игровым константам.

DECLARATION

public final class Game **extends** Object

METHODS

- *getCommanderAuraBonusActionPoints*
  `public int` **getCommanderAuraBonusActionPoints**`( )`
  - **Returns** - Возвращает количество дополнительных очков действия, которые получает юнит игрока в начале своего хода, если рядом с ним находится командир этого же игрока.
- *getCommanderAuraRange*
  `public double` **getCommanderAuraRange**`( )`
  - **Returns** - Возвращает максимальную дальность от командира, при которой юнит игрока получает дополнительные очки действия в начале своего хода.
- *getCommanderRequestEnemyDispositionCost*
  `public int` **getCommanderRequestEnemyDispositionCost**`( )`
  - **Returns** - Возвращает количество очков действия, необходимое командиру, для запроса в штаб.
- *getCommanderRequestEnemyDispositionMaxOffset*
  `public int` **getCommanderRequestEnemyDispositionMaxOffset**`( )`
  - **Returns** - Возвращает модуль максимально возможного отклонения каждой из координат расположения противника, полученных в результате вызова самолёта-разведчика.

- *getFieldMedicHealBonusHitpoints*
`public int` **getFieldMedicHealBonusHitpoints**`( )`

  - **Returns** - Возвращает количество очков здоровья, которое полевой медик может восполнить дружественному бойцу (исключая себя) за одно действие лечения.

- *getFieldMedicHealCost*
`public int` **getFieldMedicHealCost**`( )`

  - **Returns** - Возвращает количество очков действия, затрачиваемое полевым медиком на одно действие лечения.

- *getFieldMedicHealSelfBonusHitpoints*
`public int` **getFieldMedicHealSelfBonusHitpoints**`( )`

  - **Returns** - Возвращает количество очков здоровья, которое полевой медик может восполнить себе за одно действие лечения.

- *getFieldRationBonusActionPoints*
`public int` **getFieldRationBonusActionPoints**`( )`

  - **Returns** - Возвращает количество бонусных очков действия, полученных после употребления сухого пайка.

- *getFieldRationEatCost*
`public int` **getFieldRationEatCost**`( )`

  - **Returns** - Возвращает количество очков действия, необходимое для употребления сухого пайка.

- *getGrenadeCollateralDamage*
`public int` **getGrenadeCollateralDamage**`( )`

  - **Returns** - Возвращает урон от осколков гранаты.

- *getGrenadeDirectDamage*
`public int` **getGrenadeDirectDamage**`( )`

  - **Returns** - Возвращет урон от прямого попадания гранаты.

- *getGrenadeThrowCost*
`public int` **getGrenadeThrowCost**`( )`

  - **Returns** - Возвращает количество очков действия, необходимое для броска гранаты.

- *getGrenadeThrowRange*
`public double` **getGrenadeThrowRange**`( )`

  - **Returns** - Возвращает дальность броска гранаты.

- *getKneelingMoveCost*
`public int` **getKneelingMoveCost**`( )`

  - **Returns** - Возвращает количество очков действия, необходимое для перемещения на одну клетку бойца, находящегося в положении сидя.

- *getLastPlayerEliminationScore*
`public int` **getLastPlayerEliminationScore**`( )`

  - **Returns** - Количество дополнительных баллов (помимо баллов, начисляемых за урон) за уничтожение последнего бойца противника.

- *getMedikitBonusHitpoints*
`public int` **getMedikitBonusHitpoints**`( )`

- **Returns** - Возвращает количество очков здоровья, которое боец, применивший аптечку, может восполнить дружественному бойцу (исключая себя).

- *getMedikitHealSelfBonusHitpoints*
`public int` **getMedikitHealSelfBonusHitpoints**`( )`

  - **Returns** - Возвращает количество очков здоровья, которое боец, применивший аптечку, может восполнить себе.

- *getMedikitUseCost*
`public int` **getMedikitUseCost**`( )`

  - **Returns** - Возвращает количество очков действия, необходимое для использования аптечки.

- *getMoveCount*
`public int` **getMoveCount**`( )`

  - **Returns** - Возвращает длительность игры в ходах — циклах, когда каждый боец на игровом поле совершит ход один раз. Юнит считается совершившим ход, если закончились отведённые ему очки действия либо ход был передан принудительно (`ActionType.END_TURN`). В некоторых случаях игра может закончиться ранее.

- *getPlayerEliminationScore*
`public int` **getPlayerEliminationScore**`( )`

  - **Returns** - Количество дополнительных баллов (помимо баллов, начисляемых за урон) за уничтожение последнего бойца игрока. Неприменимо в случае начисления баллов, оговорённом в комментарии к методу `getLastPlayerEliminationScore()`.

- *getProneMoveCost*
`public int` **getProneMoveCost**`( )`

  - **Returns** - Возвращает количество очков действия, необходимое для перемещения на одну клетку бойца, находящегося в положении лёжа.

- *getScoutStealthBonusNegation*
`public double` **getScoutStealthBonusNegation**`( )`

  - **Returns** - Возвращает величину бонуса к маскировке юнита, которую разведчик может проигнорировать.

- *getSniperKneelingShootingRangeBonus*
`public double` **getSniperKneelingShootingRangeBonus**`( )`

  - **Returns** - Возвращает бонус к дальности стрельбы снайпера, находящегося в положении сидя.

- *getSniperKneelingStealthBonus*
`public double` **getSniperKneelingStealthBonus**`( )`

  - **Returns** - Возвращает бонус к маскировке снайпера, находящегося в положении сидя.

- *getSniperProneShootingRangeBonus*
`public double` **getSniperProneShootingRangeBonus**`( )`

  - **Returns** - Возвращает бонус к дальности стрельбы снайпера, находящегося в положении лёжа.

- *getSniperProneStealthBonus*
`public double` **getSniperProneStealthBonus**`( )`

  - **Returns** - Возвращает бонус к маскировке снайпера, находящегося в положении лёжа.

- *getSniperStandingShootingRangeBonus*
`public double` **getSniperStandingShootingRangeBonus**`( )`

- **Returns** - Возвращает бонус к дальности стрельбы снайпера, находящегося в положении стоя.

- *getSniperStandingStealthBonus*
`public double` **getSniperStandingStealthBonus**`( )`

  - **Returns** - Возвращает бонус к маскировке снайпера, находящегося в положении стоя.

- *getStanceChangeCost*
`public int` **getStanceChangeCost**`( )`

  - **Returns** - Возвращает количество очков действия, необходимое для смены стойки бойца на один уровень.

- *getStandingMoveCost*
`public int` **getStandingMoveCost**`( )`

  - **Returns** - Возвращает количество очков действия, необходимое для перемещения на одну клетку бойца, находящегося в положении стоя.

- *getTrooperDamageScoreFactor*
`public double` **getTrooperDamageScoreFactor**`( )`

  - **Returns** - Возвращает коэффициент начисления баллов за урон, нанесённый бойцу противника. В случае дробных величин значение всегда округляется вниз.

- *getTrooperEliminationScore*
`public int` **getTrooperEliminationScore**`( )`

  - **Returns** - Количество дополнительных баллов (помимо баллов, начисляемых за урон) за уничтожение бойца противника. Неприменимо в случаях начисления баллов, оговорённых в комментариях к методам `getLastPlayerEliminationScore()` и `getPlayerEliminationScore()`.

4.1.7 CLASS **Move**

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

DECLARATION

public final class Move

extends Object

METHODS

- *getAction*
`public ActionType` **getAction**`( )`

  - **Returns** - Возвращает текущее действие бойца.

- *getDirection*
`public Direction` **getDirection**`( )`
  - **Returns** - Возвращает текущее направление действия бойца.

- *getX*
`public int` **getX**`( )`
  - **Returns** - Возвращает текущую абсциссу цели действия.

- *getY*
`public int` **getY**`( )`
  - **Returns** - Возвращает текущую ординату цели действия.

- *setAction*
`public void` **setAction**`( ActionType` **action** `)`
  - **Usage**
    - Устанавливает действие бойца.
  - **Parameters**
    - `action` - Действие бойца.

- *setDirection*
`public void` **setDirection**`( Direction` **direction** `)`
  - **Usage**
    - Устанавливает направление действия бойца, если это необходимо. При обработке симулятором игры является более приоритетным, чем поля `x` и `y`.
  - **Parameters**
    - `direction` - Направление действия.

- *setX*
`public void` **setX**`( int` **x** `)`
  - **Usage**
    - Задаёт абсциссу цели действия. Значение будет проигнорировано, если установлено поле `direction`.
  - **Parameters**
    - `x` - Абсцисса цели.

- *setY*
`public void` **setY**`( int` **y** `)`
  - **Usage**
    - Задаёт ординату цели действия. Значение будет проигнорировано, если установлено поле `direction`.
  - **Parameters**
    - `y` - Ордината цели.

4.1.8 CLASS **Player**

Содержит данные о текущем состоянии игрока.

DECLARATION

public final class Player

extends Object

METHODS

- *getApproximateX*
`public int` **getApproximateX**`( )`
  - **Returns** - Возвращает примерное значение координаты X расположения противника согласно данным, полученным в результате вылета самолёта-разведчика, или `-1` в следующих случаях: вылет не производился; боец, совершающий ход, не является командиром; у данного игрока не осталось в живых ни одного бойца.
- *getApproximateY*
`public int` **getApproximateY**`( )`
  - **Returns** - Возвращает примерное значение координаты Y расположения противника согласно данным, полученным в результате вылета самолёта-разведчика, или `-1` в следующих случаях: вылет не производился; боец, совершающий ход, не является командиром; у данного игрока не осталось в живых ни одного бойца.
- *getId*
`public long` **getId**`( )`
  - **Returns** - Возвращает уникальный идентификатор игрока.
- *getName*
`public String` **getName**`( )`
  - **Returns** - Возвращает имя игрока.
- *getScore*
`public int` **getScore**`( )`
  - **Returns** - Возвращает текущее количество баллов, набранных игроком.
- *isStrategyCrashed*
`public boolean` **isStrategyCrashed**`( )`
  - **Returns** - Возвращает специальный флаг — показатель того, что стратегия игрока «упала». Более подробную информацию можно найти в документации к игре.

4.1.9 CLASS **Trooper**

Класс, определяющий бойца. Содержит также все свойства юнита.

Declaration

public final class Trooper

extends Unit

Methods

- *getActionPoints*
  `public int` **getActionPoints**`( )`
  - **Returns** - Возвращает текущее количество очков действия у бойца.

- *getDamage*
  `public int` **getDamage**`( )`
  - **Returns** - Возвращает урон одного выстрела бойца в данной стойке.

- *getDamage*
  `public int` **getDamage**`( TrooperStance  stance )`
  - **Parameters**
    - `stance` - Стойка бойца для подсчёта урона.
  - **Returns** - Возвращает урон одного выстрела бойца в указанной стойке.

- *getHitpoints*
  `public int` **getHitpoints**`( )`
  - **Returns** - Возвращает текущее количество очков здоровья у бойца.

- *getInitialActionPoints*
  `public int` **getInitialActionPoints**`( )`
  - **Returns** - Возвращает количество очков действия, которое даётся бойцу в начале хода.

- *getKneelingDamage*
  `public int` **getKneelingDamage**`( )`
  - **Returns** - Возвращает урон одного выстрела бойца, находящегося в положении сидя.

- *getMaximalHitpoints*
  `public int` **getMaximalHitpoints**`( )`
  - **Returns** - Возвращает максимальное количество очков здоровья у бойца.

- *getPlayerId*
  `public long` **getPlayerId**`( )`
  - **Returns** - Возвращает идентификатор игрока, в команду которого входит боец.

- *getProneDamage*
  `public int` **getProneDamage**`( )`
  - **Returns** - Возвращает урон одного выстрела бойца, находящегося в положении лёжа.

- *getShootCost*
  `public int` **getShootCost**`( )`
  - **Returns** - Возвращает количество очков действия, необходимое бойцу для совершения выстрела.

- *getShootingRange*
  `public double` **getShootingRange**`( )`
  - **Returns** - Возвращает дальность стрельбы бойца.

- *getStance*
  `public TrooperStance` **getStance**`( )`
  - **Returns** - Возвращает стойку бойца.

- *getStandingDamage*
  `public int` **getStandingDamage**`( )`
  - **Returns** - Возвращает урон одного выстрела бойца, находящегося в положении стоя.

- *getTeammateIndex*
  `public int` **getTeammateIndex**`( )`
  - **Returns** - Возвращает 0-индексированный номер бойца в команде.

- *getType*
  `public TrooperType` **getType**`( )`
  - **Returns** - Возвращает тип бойца.

- *getVisionRange*
  `public double` **getVisionRange**`( )`
  - **Returns** - Возвращает дальность обзора бойца.

- *isHoldingFieldRation*
  `public boolean` **isHoldingFieldRation**`( )`
  - **Returns** - Возвращает `true`, если и только если данный боец несёт с собой сухой паёк.

- *isHoldingGrenade*
  `public boolean` **isHoldingGrenade**`( )`
  - **Returns** - Возвращает `true`, если и только если данный боец несёт с собой гранату.

- *isHoldingMedikit*
  `public boolean` **isHoldingMedikit**`( )`
  - **Returns** - Возвращает `true`, если и только если данный боец несёт с собой аптечку.

- *isTeammate*
  `public boolean` **isTeammate**`( )`
  - **Returns** - Возвращает `true`, если и только если данный боец является дружественным.

4.1.10 CLASS **TrooperStance**

Стойка бойца.

DECLARATION

public final class TrooperStance

extends Enum

FIELDS

- public static final TrooperStance PRONE
  - Положение лёжа.
- public static final TrooperStance KNEELING
  - Положение сидя.
- public static final TrooperStance STANDING
  - Положение стоя.

METHODS

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

4.1.11 CLASS **TrooperType**

Тип бойца.

DECLARATION

public final class TrooperType

extends Enum

FIELDS

- public static final TrooperType COMMANDER

  - Командир. Одним своим присутствием командир способен поднять боевой дух солдат. Все юниты игрока в радиусе 5 клеток от командира получают в начале своего хода на 2 очка действия больше, чем обычно. Не распространяется на самого командира.

- public static final TrooperType FIELD_MEDIC
  - Полевой медик. Медик способен залечивать раны бойцов прямо на поле боя. Потратив одно очко действия, он может восполнить 5 единиц здоровья юниту, находящемуся на выбранной соседней (по вертикали или горизонтали) клетке, или 3 единицы здоровья себе. Здоровье юнита не может стать больше 100.

- public static final TrooperType SOLDIER
  - Штурмовик. Для прорыва хорошо укреплённой линии обороны противника штурмовик оснащён стальным нагрудником, который способен принять на себя некоторое количество урона — начальное количество очков здоровья у штурмовика на 20 больше, чем у любого другого юнита. Однако штурмовик может быть излечен полевым медиком или аптечкой первой помощи только до уровня в 100 очков здоровья.

- public static final TrooperType SNIPER
  - Снайпер. Снайпер в совершенстве владеет искусством маскировки. В процессе проверки видимости снайпера, находящегося в положении сидя, эффективный радиус обзора юнитов противника считается меньше табличного на 0.5 клетки. Если снайпер находится в положении лёжа, то штраф составляет 1 клетку. Также в положении сидя дальность стрельбы снайпера увеличивается на 1, а в положении лёжа — на 2.

- public static final TrooperType SCOUT
  - Разведчик. Намётанный глаз разведчика позволяет отличать замаскированные цели: разведчик не получает штрафа к дальности обзора при обнаружении снайперов противника. У разведчика на 2 очка действия больше, чем у других бойцов, однако он привык действовать самостоятельно и поэтому не получает дополнительных очков действия рядом с командиром.

METHODS

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

4.1.12 CLASS **Unit**

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

DECLARATION

---

public abstract class Unit **extends** Object

- *getDistanceTo*
  `public double` **getDistanceTo**`( int x, int y )`
  - **Parameters**
    - `x` - X-координата клетки.
    - `y` - Y-координата клетки.
  - **Returns** - Возвращает расстояние между центрами указанной клетки и текущей клетки объекта.

- *getDistanceTo*
  `public double` **getDistanceTo**`( Unit unit )`
  - **Parameters**
    - `unit` - Объект, до центра которого необходимо определить расстояние.
  - **Returns** - Возвращает расстояние между центрами клеток текущего и указанного объекта.

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

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

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

4.1.13 CLASS **World**

Содержит описание игрового мира и позволяет получить списки различных юнитов, присутствующих на поле боя.

DECLARATION

public final class World **extends** Object

METHODS

- *getBonuses*
  `public Bonus[]` **getBonuses**`( )`

- **Returns** - Возвращает список видимых юнитами игрока бонусов (в случайном порядке). Объекты, задающие бонусы, пересоздаются перед каждым вызовом `Strategy.move()`.

- *getCells*
`public CellType[][] getCells( )`
  - **Returns** - Возвращает двумерный массив типов клеток игрового поля, где первое измерение — это координата X, а второе — Y.

- *getCellVisibilities*
`public boolean[] getCellVisibilities( )`
  - **Usage**
    - Возвращает массив досягаемости различных клеток игрового мира из других клеток без учёта расстояния между ними.
    Боец в клетке `(objectX, objectY)` является досягаемым для бойца в клетке `(viewerX, viewerY)`, если и только если `cellVisibilities[viewerX * height * width * height * stanceCount + viewerY * width * height * stanceCount + objectX * height * stanceCount + objectY * stanceCount + minStanceIndex]` равно `true`, где `height` — высота мира в клетках, `width` — ширина мира в клетках, `stanceCount` — количество различных значений перечисления `TrooperStance`, а `minStanceIndex` — минимальный из индексов стоек двух указанных бойцов (стойка `TrooperStance.PRONE` имеет индекс `0`).
  - **Returns** - Возвращает массив досягаемости.

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

- *getMoveIndex*
`public int getMoveIndex( )`
  - **Returns** - Возвращает номер текущего хода.

- *getPlayers*
`public Player[] getPlayers( )`
  - **Returns** - Возвращает список игроков (в случайном порядке). Объекты, задающие игроков, пересоздаются перед каждым вызовом `Strategy.move()`.

- *getTroopers*
`public Trooper[] getTroopers( )`
  - **Returns** - Возвращает список видимых юнитами игрока бойцов (в случайном порядке), включая бойца стратегии, вызвавшей этот метод. Объекты, задающие бойцов, пересоздаются перед каждым вызовом `Strategy.move()`.

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

- *isVisible*
`public boolean isVisible( double maxRange, int viewerX, int viewerY, TrooperStance viewerStance, int objectX, int objectY, TrooperStance objectStance )`
  - **Usage**

    * Метод проверяет, является ли юнит, находящийся в клетке с координатами (`objectX`, `objectY`) в стойке `objectStance`, досягаемым для юнита, находящегося в клетке с координатами (`viewerX`, `viewerY`) в стойке `viewerStance`. Может использоваться как для проверки видимости, так и для проверки возможности стрельбы.
      При проверке видимости бонуса его высота считается равной высоте бойца в стойке `TrooperStance.PRONE`.

- **Parameters**
    * `maxRange` - Дальность обзора/стрельбы наблюдающего юнита (`viewer`).
    * `viewerX` - X-координата наблюдающего юнита (`viewer`).
    * `viewerY` - Y-координата наблюдающего юнита (`viewer`).
    * `viewerStance` - Стойка наблюдающего юнита (`viewer`).
    * `objectX` - X-координата наблюдаемого юнита (`object`).
    * `objectY` - Y-координата наблюдаемого юнита (`object`).
    * `objectStance` - Стойка наблюдаемого юнита (`object`).
- **Returns** - Возвращает `true`, если и только если наблюдаемый юнит (`object`) является досягаемым для наблюдающего юнита (`viewer`).

Глава 5

Package <none>

Package Contents *Page*

Interfaces

Стратегия — интерфейс, содержащий описание методов искусственного интеллекта бойца.

5.1 Interfaces

5.1.1 INTERFACE **Strategy**

Стратегия — интерфейс, содержащий описание методов искусственного интеллекта бойца. 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**`( Trooper` **self**`, World` **world**`, Game` **game**`, Move` **move** `)`
  - **Usage**
    - Основной метод стратегии, осуществляющий управление бойцом. Многократно вызывается для бойца, совершающего ход, до тех пор, пока у него есть очки действия. Стратегия может завершить ход принудительно, установив `Move.action` в значение `END_TURN`.
  - **Parameters**
    - `self` - Боец, которым данный метод будет осуществлять управление.
    - `world` - Текущее состояние мира.
    - `game` - Различные игровые константы.
    - `move` - Результатом работы метода является изменение полей данного объекта.