Санкт-Петербургский государственный университет
Математическое обеспечение и администрирование информационных систем

Кафедра Системного Программирования

Кулеш Сергей Павлович

Создание бота для командной тактико-стратегической игры Dota 2

Дипломная работа

Научный руководитель:
ст. преп. Зеленчук И. В.

Рецензент:
магистр Кокорев А. Д.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY
Software and Administration of Information Systems

Software Engineering Chair

Sergey Kulesh

Creation of bot for tactical and strategic team game Dota2

Graduation Thesis

Scientific supervisor:
Senior lecturer Ilya Zelenchuk

Reviewer:
Master Andrey Kokorev

Saint-Petersburg
2019

Оглавление

1. Введение **4**

2. Цель работы **6**

2.1. Цель 6

2.2. Задачи 6

3. Обзор **7**

3.1. Игра 7

3.2. API 9

3.3. Существующие решения 10

3.3.1. Боты из мастерской и стандартные боты 10

3.3.2. OpenAI 10

3.4. Герой 13

4. Решение **14**

4.1. Бот без использования машинного обучения 14

4.2. Бот с использованием машинного обучения 14

4.2.1. Bodyblock 15

5. Заключение **21**

Список литературы **22**

1. Введение

Киберспорт, под которым понимают соревнования на основе видеоигр, зародился в 1997 году, вместе с основанием первой киберспортивной лиги CPL. Это довольно молодая, но очень быстрорастущая, особенно в последнее время, отрасль. С 7 июня 2016 года компьютерный спорт внесен в реестр официальных видов спорта Российской Федерации. Согласно "Newzoo" [3] к 2019 году данная отрасль принесет выручку в 1,1 млрд. долларов при росте в 26,7 процентов, а аудитория киберспорта достигнет отметки в 454 млн. человек.

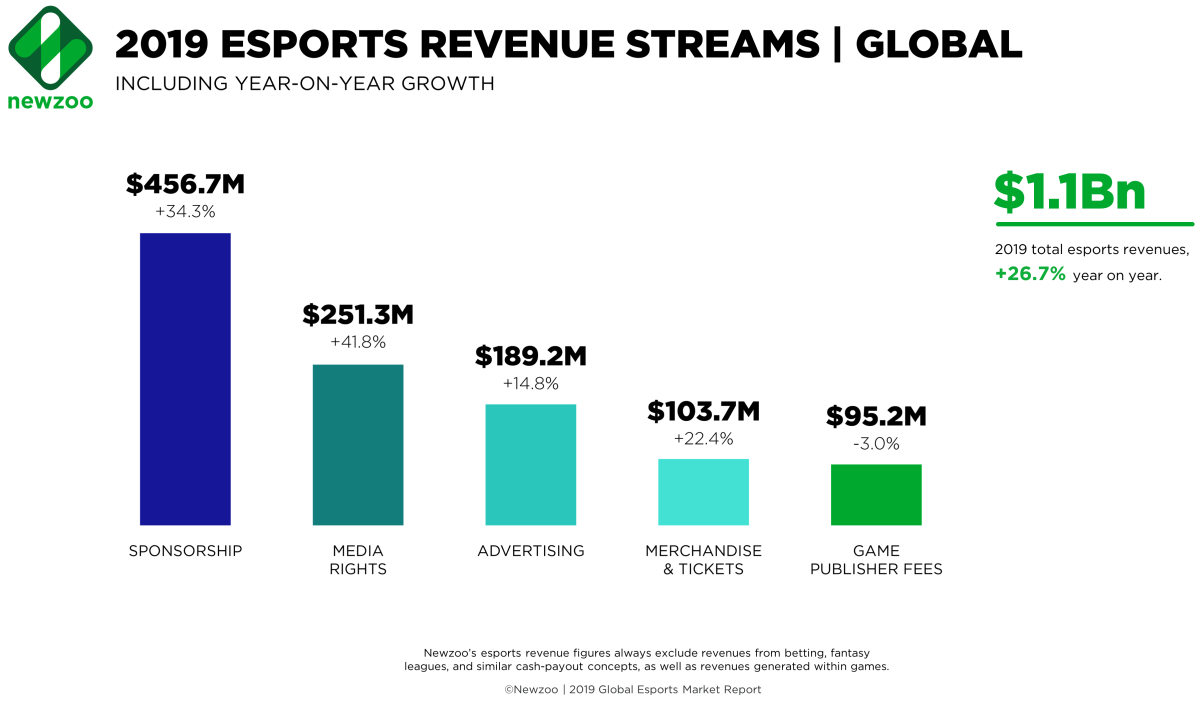

Рис. 1: Newzoo Analytics [5]

Dota 2 [2] - одна из наиболее популярных киберспортивных дисциплин, созданная компанией Valve, с ежедневным online около 600 тысяч человек (максимальный онлайн составляет 1.29 млн человек). По состоянию на 2018 год призовой фонд главного турнира "The International" составил свыше 25 млн. долларов, а суммарный призовой фонд по этой дисциплине давно превысил 100 млн. долларов. К 2019 году "The International" планирует собрать призовой фонд в 30 млн. долларов.

При таких больших цифрах тренировочная система не обладает должной эффективностью. Игроки тренируются исключительно друг против друга, что не способствует достижению желаемого результата. Не существует так называемого “тренажера”, на котором игроки могли бы отработать свою технику игры, командные взаимодействия и тактики. Это как если бы у футболистов не было тренировок, кроме матчей, - такое даже трудно представить. Бот - это программа-робот, управляемая компьютером, имитирующая партнеров (противников) в сетевой игре. Учитывая, что бот может выполнять команды без ошибок, не испытывая эмоций, которые могут влиять на уровень игры, тренироваться против него - эффективнее, нежели против живого игрока.

В игре реализованы стандартные боты различной сложности, однако их уровня игры не достаточно для плодотворных тренировок (он оценивается как уровень среднестатистического игрока), более того, некоторые важные аспекты игры ими не учитываются. Таким образом, создание компьютерной программы, способной играть и выигрывать стандартного бота высокой сложности в Dota 2, является перспективной и востребованной задачей.

2. Цель работы

2.1. Цель

Целью данной работы является создание программы (бота) с элементами искусственного интеллекта, способного играть и выигрывать у стандартного бота высокой сложности на центральной линии в режиме игры ”1 на 1”, на котором можно оттачивать отдельные элементы игры.

2.2. Задачи

Для достижения данной цели были поставлены следующие задачи:

- изучить предметную область и методы взаимодействия с игровым процессом;
- сделать обзор существующих решений и подходов к реализации бота как механизма для тренировок;
- выбрать решение для дальнейшей работы;
- написать бота без использования машинного обучения, исследовать результат;
- развить модель бота путем внедрения машинного обучения в отдельные области игрового процесса.

3. Обзор

3.1. Игра

«Dota» есть аббревиатура: «Defence of the ancients». «Древними» называются главные здания двух противоборствующих сторон – «светлых» и «темных». В начале игры игроку предлагается выбрать героя (одного из 118) и принять участие в битве за одну из сторон, чтобы победить, разрушив трон вражеской фракции. На каждой из сторон может быть по пять героев, управляемых игроками. Стандартная игра протекает в формате «пять на пять», но нас интересует конкретный режим игры - “1 на 1”.

Рис. 2: Схематичная карта игры

В этом режиме игроки сражаются на центральной линии. Задача: убить противника дважды или уничтожить первую башню. Для модели бота выбран один из 118 героев “Shadow Fiend” [9], как самый популярный представитель данного режима.

После выбора героя пойдет обратный отсчет времени. Ровно в 0:00,

и каждые последующие 30 секунд возле бараков на центральной линии будут появляться группы «крипов» (npc), управляемые компьютером. Они начинают двигаться по линии в сторону противника, и в определенном месте встречаются с вражескими крипами.

Персонажу требуется игровая валюта для приобретения артефактов. Основной способ ее добычи – добивание крипов (фарм), а именно: когда у крипа здоровье будет близко к нулевому значению, герой наносит ему добивающий удар и получает деньги за убийство. Добивание своих и вражеских крипов одинаково по сути, но ведет к разным результатам. Союзный крип лишает денег и опыта соперника, вражеский - заработок валюты и опыта. Но основная цель состоит не в добивания крипов и наращивании преимущества над противником (это средства ее достижения), а в победе над ним.

Что касается технической стороны - Dota существует с 2005 года и регулярно получает обновления, которые постоянно меняют ее семантику. Среднестатистическая игра длится 45 минут при 30 кадрах в секунду (получается 80000 тиков - гиганская масса данных), причем отдельные действия могут кардинально переворачивать ход игры. Также игра осложнена неполными данными, так как юниты могут видеть лишь ограниченную область вокруг них, а значит необходимо моделировать возможные действия противника.

3.2. API

В 2016 году компания Valve выложили в открытый доступ API [1]. Оно представляет из себя список скриптовых функций и переменных на lua, для взаимодействия с игровым процессом. Реализовано все таким образом, что нет необходимости эмулировать движение мышки или просматривать экранные пиксели - достаточно подать запрос на состояние игры или отдать команду персонажу напрямую. При этом можно действовать лишь в рамках правил игры и команды не могут подаваться не подконтрольному юниту. Более того, для ботов стоит ограниченное время реакции, схожее с человеческим. В дополнении существует внутренний код на C++, который реализует стандартных ботов игры, так что допускается как переписывание лишь некоторых функций, так и написание бота с нуля.
Существует 3 уровня организации:

- team level отвечает за то, какое из глобальных действий хочет осуществить команда в данным момент времени, но не предписывает никаких действий, отражая лишь направленность командного взаимодействия, например, фарм или защита одной из линий;
- mode level отвечает за то, какое из глобальных действий хочет осуществить конкретный бот в данный момент времени, например фарм, или попытка убить противника;
- action level отвечает за действия бота в текущий момент времени, например, атаковать, использовать способность, идти.

Team level является руководством для общих действий команды, каждый бот вычисляет собственный mode level, который сочетается с team level и определяет action level. Однако, для режима "1 на 1" team level не нужен, достаточно пользоваться лишь двумя остальными уровнями, или полностью переписать логику поведения бота.

3.3. Существующие решения

3.3.1. Боты из мастерской и стандартные боты

Стандартные боты реализованы в игре на разных уровнях сложности:

- пассивные, легкие и средние нас не интересуют, так как имеют слишком замедленное время реакции и множество других ограничений;
- сложные имеют случайную задержку на использование предметов и способностей в 0.1 - 0.2 секунды, что близко к реакции человека, которая составляет 0.2 секунды [11];
- нечестные боты так же не рассматриваются, потому что их время реакции ощутимо меньше человеческого, получение опыта и игровой валюты для них увеличено на 25 процентов.

Таким образом, на рассмотрении остался только бот с уровнем сложности "сложный", который играет на уровне среднестатистического игрока, однако, стандартные боты реализованы на языке C++ и код находится в закрытом доступе, поэтому они могут быть рассмотрены лишь в качестве метрики эффективности моего бота.

Мастерская Steam - это база плагинов, иными словами, - место, куда пользователи могут добавлять созданный ими контент, который можно будет добавить в игру [10]. В моем случае - это хранилище любительских ботов. Я выделил всего три подходящих под мою задачу: "[JPBot] 1vs1 SF mid", "StrawberryBot — 1vs1 Mid SF Bot", "Mid Only Solo". Все они представляют собой стандартных ботов с дописанным функционалом, и так же будут рассмотрены лишь в качестве метрики эффективности.

3.3.2. OpenAI

В 2017 году компания OpenAI представила первого в мире бота для Dota 2, который смог на главном турнире одолеть самого известного

профессионального игрока Даниила Ишутина (Dendi) [6]. В 2018 году эта же компания представила команду ботов, но обыграть команду профессиональных игроков тогда еще не удалось. В апреле 2019 года команде OpenAI удалось победить действующих мировых чемпионов - команду OG, но с ограничениями на выбор героев [7].

	OPENAI 1V1 BOT	OPENAI FIVE
CPUs	60,000 CPU cores on Azure	128,000 preemptible CPU cores on GCP
GPUs	256 K80 GPUs on Azure	256 P100 GPUs on GCP
Experience collected	~300 years per day	~180 years per day (~900 years per day counting each hero separately)
Size of observation	~3.3 kB	~36.8 kB
Observations per second of gameplay	10	7.5
Batch size	8,388,608 observations	1,048,576 observations
Batches per minute	~20	~60

Рис. 3: Ресурсы OpenAI

OpenAI реализован через обучение с подкреплением [12] и учится на самостоятельной игре (начиная со случайных весов), которая обеспечивает естественную учебную программу для изучения окружающей среды. Обучение с подкреплением - это область машинного обучения, которая состоит из агента и среды, взаимодействуя с которой, агент получает подкрепление посредством положительного или отрицательного вознаграждения. В зависимости от этих подсказок, агент меняет свое поведение. Конечная цель этого процесса — получить как можно

большую награду, или, по-другому, достигнуть тех действий, которые поставили перед агентом.

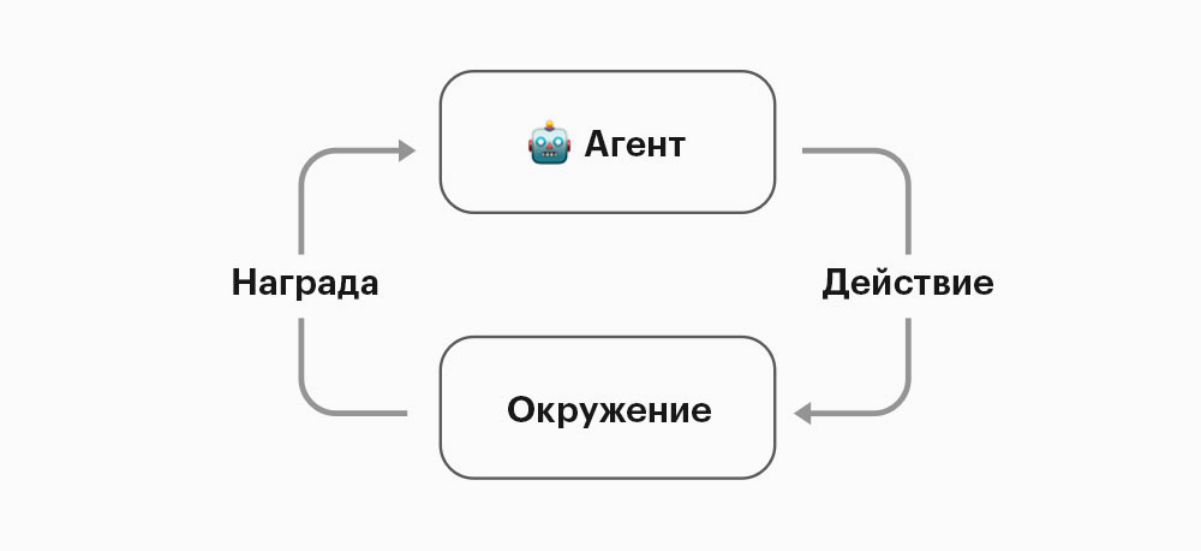

Рис. 4: Reinforcement learning [4]

В первых играх герои бесцельно ходят по карте. После нескольких часов тренировок появляется некоторая логика в действиях. Через несколько дней они последовательно применяют базовые человеческие стратегии. И с дальнейшим обучением они становятся способны на игру высокого уровня.

Существующее решение от OpenAI требует колоссальных ресурсов, доступ к которым подразумевает серьезное финансирование и целую команду специалистов. К сожалению, ни доступа к коду, ни возможности протестировать бота нет. Вся информация находится в закрытом доступе. Для достижения эффективного результата планировалось постепенное внедрение машинного обучения в некоторые области игрового процесса. Таким образом, первичной задачей было написание алгоритмов поведения бота без использования ML, исследование результатов, и последующее развитие модели, путем введения ML.

3.4. Герой

Самый популярный представить режима игры ”1vs1 mid only” - персонаж ”Shadow Fiend”. Так же данный герой является классическим центральным игроком - соревнования на лучшего игрока первой позиции команды проходят именно на нем. Именно поэтому я выбрал данного героя для написания бота. Рассмотрим его подробнее.

Shadow Fiend — персонаж дальнего боя. Благодаря своей пассивной способности, Necromastery , которая крадет души убитых противников, увеличивая атаку на 2 урона за каждую душу, на этом герое и проводят соревнования в режиме ”1vs1” - успех на линии для данного героя сильно зависит от добивания и отнимания крипов. При изначальной слабости, в процессе накопления душ SF становится крайне сильным и опасным на линии. Но отсутствие большого запаса здоровья требует аккуратной и бдительной игры.

4. Решение

4.1. Бот без использования машинного обучения

Мною был реализован бот для режима 1 на 1, без использования ML. При создании я основывался на личном игровом опыте, что сильно упростило задачу, так как я знал все тонкости игры, что бот должен делать, как и когда. Процесс получился долгий и трудоемкий - общий объем кода занимает около 2000 строк, реализован большой спектр логики поведения бота в различных ситуациях игрового процесса. Что же касается эффективности, я выбрал метрику сравнения со стандартным ботом, а так же с ботами из мастерской, подходящими под поставленную задачу, а именно - мой бот вышел победителем против каждого из представленных соперников в серии до двух выигрышей в матчах.

4.2. Бот с использованием машинного обучения

Когда встал вопрос о внедрении ML возникла проблема - как получать данные из игры и взаимодействовать с ней из вне? Здесь мне на помощь пришли ребята из компании Jet Brains с их готовой работой [8], цель которой - использовать один из самых популярных и быстроразвивающихся разделов машинного обучения — обучение с подкреплением, на какой-нибудь очень сложной модели - на Dota 2. Более подробно целью было максимизация числа атак бота, что можно соотнести с моей задачей лишь в некотором приближении. Работа JB помогла мне структурой, которую они создали для обучения, правда, потребовалось довольно много времени, чтобы настроить ее под свое устройство, и адаптировать под задачу.

Клиент Dota 2 существует под все популярные платформы (Windows, macOS и Linux). Однако, обычно обучение происходит в специальной среде, которая создается отдельно. Для Dota 2 нет такой среды. Ребята из JB сделали ее и состоит она из 3 частей:

- первая часть — скрипт бота, взаимодействуя с клиентом игры, он

собирает нужную нам информацию о состоянии игры и отправляет ее как json файл на сервер;

- вторая часть — обертка, оформленная в виде сервера, который запускает игру и принимает информацию от скрипта: Pyautogui отвечает за запуск и рестарт игры, а Flask сервер осуществляет общение с lua-вставкой;

- третья часть содержит в себе алгоритм обучения, который выбирает действия, получает награду и следующее состояние от сервера, тем самым улучшая свое поведение.

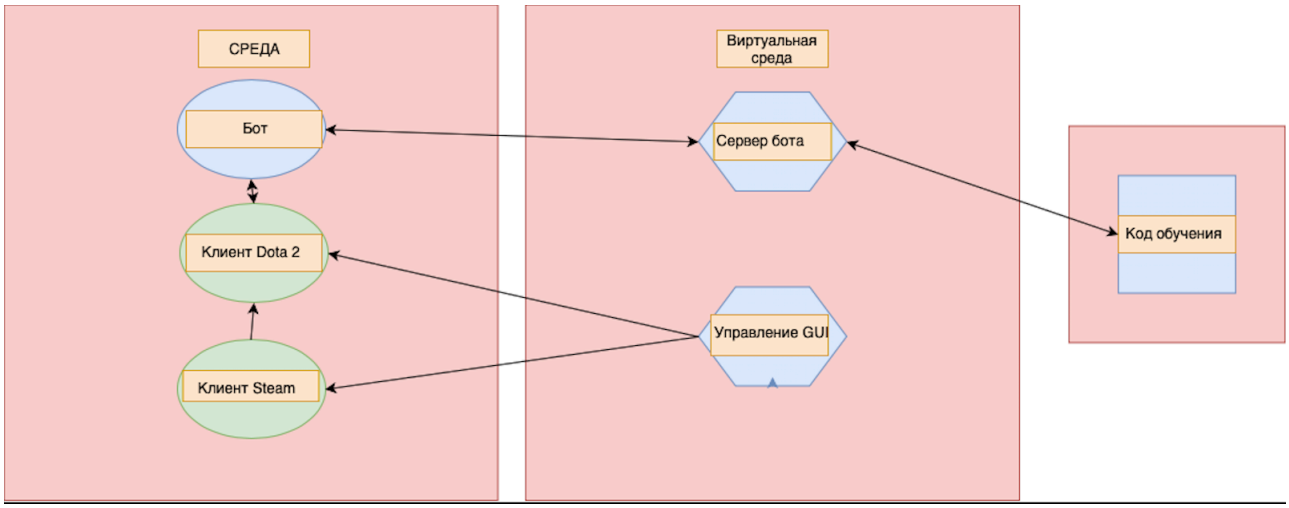

Рис. 5: Структура для обучения

4.2.1. Bodyblock

Передо мной встала задача, с чего начать обучение? Наиболее простым и очевидным было начать с так называемого Bodyblock-а, тем более OpenAI так же начинали с него, и эту часть игры можно тренировать непосредственно с ботом.

Более подробно - это первое, что необходимо сделать боту в начале игры, чтобы получить преимущество над противником. В начале игры и последующие каждые 30 секунд возле бараков каждой из команд появляются крипы, которые встречаются по центру карты. Задача состоит в том, чтобы максимально замедлить движение крипов персонажем,

чтобы они пересеклись ближе к союзной башне, в результате чего будет получено преимущество. Более простым языком: нужно моделькой героя замедлить движение 4 моделек крипов насколько это возможно, заслоняя им проход. Мною был реализован Bodyblock без использования ML, чтобы лучше понять, как применить ML, и какие данные подавать на вход.

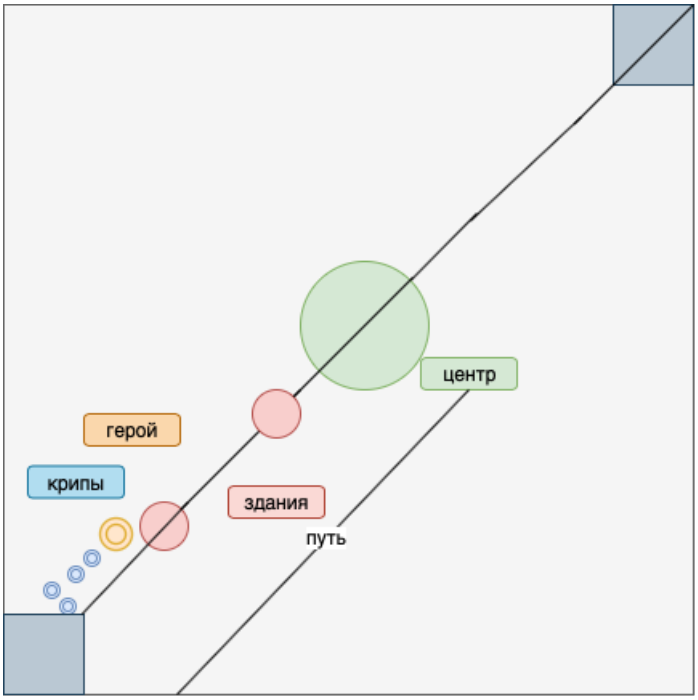

Рис. 6: Схема

Что же до метода - тут используется обучение с подкреплением, о котором я уже говорил ранее. В нашем случае агент - персонаж, его действия - перемещения по карте. Окружение - сама карта и крипы. Агент ходит по карте, и, в зависимости от состояния игры, получает награду.

Я решил использовать известный алгоритм Q-learning [13], так как уже использовал его при обучении бота для игры "крестики-нолики". Алгоритм заключается в заполнении целевой функции Q, принимающей два параметра - состояние и действие из этого состояния. Значения целевой функции - потенциал выбрать определенное действие, будучи в определенном состоянии, суть: заполнить значения этой функции от всех подаваемых входов. При этом существует политика выбора

действий. Самая распространенная - epsilon-greedy policy: с маленькой epsilon вероятностью выбираем случайное действие, в противном случае берем максимум целевой функции по всем возможным действиям из данного состояния.

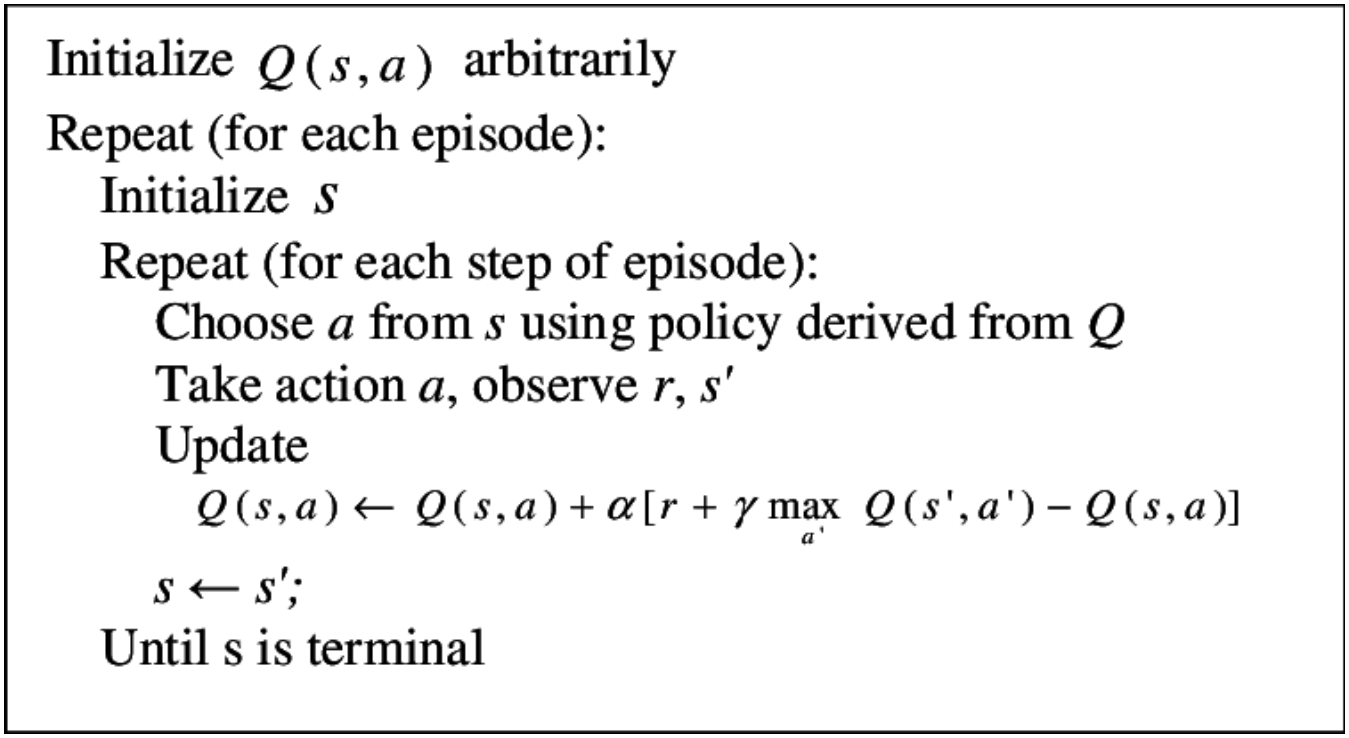

Рис. 7: Q-learning [13]

α — это фактор обучения. Чем он выше, тем сильнее агент доверяет новой информации. γ — это фактор дисконтирования. Чем он меньше, тем меньше агент задумывается о выгоде от будущих своих действий.

Однако в нашем случае существует ряд проблем. Положение на карте задается как вектор из 3 координат (одну координату можно опустить, она не влияет на процесс). Перемещения бота - движение в определенную локацию, так же задается вектором. В итоге пространство действий получается не дискретным, что сильно усложняет задачу. Так как действия бота явно должны зависеть от положения крипов, я решил свести все возможные варианты всего к двум - либо бот стоит на месте, либо движется в направлении, зависящем от текущего положения крипов. Так же эффективным улучшением будет, если бот не будет стоять на месте, когда крипы подошли слишком близко, и в противовес этому - персонажу не следует двигаться, если он слишком оторвался

от крипов - это будет нашей политикой выбора действий в случае, если мы пришли в новое состояние.

Что для нас состояние? Очевидно - это местоположение героя и четырех крипов - каждое задается двумя четырех-значными числами с плавающей точкой. Если в формировании состояния примут участие все эти числа, то получится слишком много вариантов. Исследовав игровой процесс я понял, что достаточно вести лишь первые 3 цифры в координатах, и всего у двух крипов в паре с персонажем (ближайших к нему). Это существенно сокращает множество состояний.

С наградой все просто: нам нельзя допустить, чтобы крипы обошли бота. В случае, если расстояние до центра у персонажа больше, нежели у какого-нибудь крипа - мы проиграли, иначе все хорошо.

Соединив все в единый алгоритм, мы получили рабочий bodyblock, результаты которого доступны по ссылкам:

- `https://www.youtube.com/watch?v=itLYZb2vLbo`;
- `https://www.youtube.com/watch?v=5-60psYMoqE`;
- `https://www.youtube.com/watch?v=wx5duLkTSeQ`;
- сравнение с OpenAI: `https://www.youtube.com/watch?v=UiD1hw8-B1g`.

По метрике качества: если не мешать крипам, то они встретятся в центре карты по истечении 18 секунд от старта игры. В случае bodyblock-а с одной стороны первый контакт будет в диапазоне 22-25 секунд в более выгодной позиции у союзной башни, в результате чего будет получено преимущество по игре.

Рис. 8: Преимущество

На рисунке выше видно, что союзная башня атакует вражеских крипов, а до башни противника нас отделяет существенное расстояние. Это дает превосходство по позиции, а также по опыту, который будет получен раньше соперника, поскольку башня ускорит добивание крипов оппонента.

Если сравнивать с OpenAI, то я взял метрику подхода героя к центральной башне, благо в игре есть свой таймер, показывающий время с точностью до секунды. Так как весь код OpenAI находится в закрытом доступе, и нет возможности провести непосредственное тестирование, я взял записи матчей против профессиональных игроков:

- в матче против Dendi имеем результат 21 секунда от начала матча;
- в матче против Arteezy 19;
- в матче против Sumail 20;
- в матче против Pajkatt 17.

Ввиду малых объемов выборок я решил ориентироваться при сравнении со своим ботом на максимальные значения. Произвел замеры в 15 испытательных матчах своего бота и получил следующую выборку:

$$19, 21, 19, 26, 20, 20, 19, 21, 20, 20, 19, 24, 20, 21, 23$$

Математическое ожидание у данной выборки принимает значение 20.8, что сравнимо с лучшим результатом у OpenAI, однако нас интересует сравнение максимальных значений, а это 26 против 21 в пользу моего бота.

5. Заключение

В результате выполнения дипломной работы были получены следующие результаты:

- изучена предметная область и существующее решение;
- исследованы возможности API;
- реализован бот без использования машинного обучения;
- настроена инфраструктура для обучения бота;
- реализован bodyblock.

Код доступен на Github по ссылке: `https://github.com/Pechckin`.

Список литературы

[1] API for Dota 2 bot scripting. https://developer.valvesoftware.com/wiki/Dota_Bot_Scripting. Accessed: 2019-05-05.

[2] Dota 2 main page. http://www.dota2.com. Accessed: 2019-05-05.

[3] Gloval esports economy. https://newzoo.com/insights/articles/newzoo-global-esports-economy-will-top-1-billion-for-the-first- Accessed: 2019-05-05.

[4] Jet Brains work. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf. Accessed: 2019-05-05.

[5] Newzoo. https://newzoo.com. Accessed: 2019-05-05.

[6] OpenAI 1vs1. https://openai.com/blog/dota-2/. Accessed: 2019-05-05.

[7] OpenAI 5vs5. https://openai.com/five/. Accessed: 2019-05-05.

[8] Reinforcement learning. https://habr.com/ru/company/hsespb/blog/435636/. Accessed: 2019-05-05.

[9] Shadow Fiend wiki. https://dota2-ru.gamepedia.com/ShadowFiend. Accessed: 2019-05-05.

[10] Мастерская. https://steamcommunity.com/workshop. Accessed: 2019-05-05.

[11] Burkhart Fischer and E Ramsperger. Human express saccades: extremely short reaction times of goal directed eye movements, 1984.

[12] Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning, 1998.

[13] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.