МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЕТ**

**по \_\_\_\_\_\_\_\_\_\_\_\_\_\_научно-исследовательской работе\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(вид практики, тип практики)

**в \_\_\_\_ПГТУ, каф. ИиСП\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(наименование профильной организации)

**Тип практики** \_\_\_\_НИР\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Сроки практики**\_\_\_\_. .2018 – . .2019\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил обучающийся гр. ПСм-11

Колчин И. А.

(Ф.И.О.)

Руководитель практики от ПГТУ

Морозов М.Н., к.т.н., проф.,

(должность, Ф.И.О., подпись)

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Йошкар-Ола

2019

Содержание

[Постановка задачи 3](#__RefHeading___Toc30025_3354505249)

[Актуальность 4](#__RefHeading___Toc30027_3354505249)

[Почему этим нужно заниматься 4](#__RefHeading___Toc30029_3354505249)

[Кто занимался исследованиями в данной области 4](#__RefHeading___Toc30031_3354505249)

[Недостатки существующих решений и предлагаемые решения 5](#__RefHeading___Toc30033_3354505249)

[Цель и задачи 6](#__RefHeading___Toc1265_1102303937)

[Объект и предмет исследования 6](#__RefHeading___Toc1267_1102303937)

[Новизна и практическая значимость 6](#__RefHeading___Toc1269_1102303937)

[Обзор источников 7](#__RefHeading___Toc30035_3354505249)

[Deep RTS 7](#__RefHeading___Toc30037_3354505249)

[Обзор 7](#__RefHeading___Toc3932_562408801)

[Аннотация 8](#__RefHeading___Toc3934_562408801)

[Введение 9](#__RefHeading___Toc3936_562408801)

[Ранее созданные среды 11](#__RefHeading___Toc3941_562408801)

[A. Stratagus 11](#__RefHeading___Toc4157_562408801)

[B. Arcade Learning Environment 12](#__RefHeading___Toc4159_562408801)

[C. microRTS 12](#__RefHeading___Toc4161_562408801)

[D. TorchCraft 12](#__RefHeading___Toc4163_562408801)

[E. Malmo Platform 12](#__RefHeading___Toc4165_562408801)

[F. ViZDoom 13](#__RefHeading___Toc4167_562408801)

[G. DeepMind Lab 13](#__RefHeading___Toc4169_562408801)

[H. OpenAI Gym 13](#__RefHeading___Toc4171_562408801)

[I. OpenAI Universe 13](#__RefHeading___Toc4173_562408801)

[J. ELF 14](#__RefHeading___Toc4175_562408801)

[K. StarCraft II Learning Environment 14](#__RefHeading___Toc4177_562408801)

[Обучение с подкреплением в играх 14](#__RefHeading___Toc3943_562408801)

[A. TD-Gammon 14](#__RefHeading___Toc4179_562408801)

[B. AlphaGO 15](#__RefHeading___Toc4181_562408801)

[C. DeepStack 15](#__RefHeading___Toc4183_562408801)

[D. Dota 2 16](#__RefHeading___Toc4185_562408801)

[Среда Deep RTS 16](#__RefHeading___Toc3945_562408801)

[A. Цель игры 16](#__RefHeading___Toc4187_562408801)

[B. Игровая механика 18](#__RefHeading___Toc4189_562408801)

[C. Графика 19](#__RefHeading___Toc4191_562408801)

[E. Итоги 20](#__RefHeading___Toc4193_562408801)

[Эксперементы 21](#__RefHeading___Toc3947_562408801)

[B. Сравнение Deep RTS с существующими средами 22](#__RefHeading___Toc4195_562408801)

[C. Использование Q-обучения в Deep RTS 22](#__RefHeading___Toc4197_562408801)

[Заключение и будущие работы 24](#__RefHeading___Toc3949_562408801)

[Благодарность 25](#__RefHeading___Toc3951_562408801)

[Список литературы 26](#__RefHeading___Toc30039_3354505249)

# Постановка задачи

В рамках НИР нужно было начать работу над дипломом. Для этого нужно выполнить следующие задачи:

* Найти источники информации по теме диплома(статьи, дипломы, диссертации и т. д.)
* Проанализировать найденные источники
* Составить обзор на наиболее полезный источник
* Составить общую характеристику работы. В неё должна входить:
  + Актуальность
  + Цель
  + Задачи, которые нужно выполнить для достижения цели
  + Исследуемая область(объект) и направление в области(предмет)
  + Научная новизна
  + Практическая ценность

# Актуальность

## Почему этим нужно заниматься

На данный момент существует множество военных стратегий реального времени. И не все из них позволяют моделировать реалистичные или максимально приближенные к реальности битвы, в которых бы учитывались не только особенности рельефа и различных видов войск, но и другие факторы, такие как запасы еды, топлива, пути снабжения, погодные условия, сложность дорожных сетей.

Ранее студентами ПГТУ была создана игра под названием «WarOnMap», которая учитывает перечисленные выше факторы. Эта игра позволяет моделировать приближенные к реальности битвы, и кроме этого она является единственной стратегией, которая использует дорожный граф в качестве карты.

## Кто занимался исследованиями в данной области

Представленная игра является стратегией реального времени(англ. real time strategies, сокр. RTS). На данный момент наибольших успехов в применении машинного обучения в RTS добилась компания DeepMind, создав ИИ «AlphaStar» для игры Starcraft 2 Legacy of the Void. Per-Arne Andersen, Morten Goodwin и Ole-Christoffer Granmo совместно разработали среду Deep RTS для разработки и тестирования ИИ для игр жанра RTS, в которой протестировали собственный ИИ на небольшой стратегии.

Представленные стратегии отличаются от «WarOnMap» тем, что там необходимо развивать экономику, и наличием видов войск, которых не существует в реальном мире(например, самолёт который может трансформироваться в шагающего робота как в Starcraft). Кроме того представленные игры используют карты, представляющие из себя сетку на которой располагаются боевые единицы.

## Недостатки существующих решений и предлагаемые решения

Существующие решения в применении машинного обучения в играх жанра RTS позволяют эффективно управлять армией на картах типа «сетка». Но ни один из существующих ИИ не может работать с дорожным графом.

Чтобы решить выше описанную проблему нужно будет использовать алгоритм Бобкова А. С.[5] для быстрого поиска оптимального пути на дорожном графе и создать ИИ, который сможет эффективно управлять армией быстро ориентируясь на дорожном графе.

# Цель и задачи

**Целью** диссертационной работы является исследование применения современных технологий машинного обучения для создания ИИ быстро ориентирующегося на дорожном графе. Исходя из поставленной цели, в диссертационной работе решаются следующие **задачи**:

1. Найти и проанализировать различные технологии машинного обучения.
2. Выбрать подходящие для решения цели
3. Разработать API для взаимодействия ИИ с дорожным графом и игрой.
4. Создать систему для тестирования и обучения ИИ на дорожном графе.
5. Добавить систему тестирования ИИ в игру.
6. Спроектировать ИИ, который будет использовать ранее выбранные технологии.
7. Создать ИИ
8. Обучить ИИ на самом себе или реальных игроках
9. Протестировать ИИ на реальных игроках

# Объект и предмет исследования

**Объектом исследования** является машинное обучение.

**Предметом исследования** является машинное обучение в стратегиях реального времени на дорожном графе.

# Новизна и практическая значимость

**Новизна** исследования заключается в созданный ИИ будет ориентироваться на дорожном графе.

**Практическую ценность работы составляет** система разработанная для обучения и тестирования различных видов ИИ(основанных не только на технологиях машинного обучения) на дорожных графах.

# Обзор источников

Были найдены следующие источники:

1. Javier Villanueva Forner «USE OF MACHINE LEARNING TECHNIQUES IN VIDEOGAMES»[1]
2. Per-Arne Andersen, Morten Goodwin, Ole-Christoffer Granmo «Deep RTS: A Game Environment for Deep Reinforcement Learning in Real-Time Strategy Games»[2]
3. Kun Shao, Yuanheng Zhu, Member, IEEE and Dongbin Zhao, Senior Member «StarCraft Micromanagement with Reinforcement Learning and Curriculum Transfer Learning»[3]
4. Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, Nicolas Usunier «TorchCraft: a Library for Machine Learning Research on Real-Time Strategy Games»[4]
5. Hendrik Baier, Peter I. Cowling «Evolutionary MCTS for Multi-Action Adversarial Games»[6]

Для обзора была выбрана 2 статья, потому-что там описывалась среда для разработки ИИ для игра жанра стратегий реального времени. Устройство среды позволит мне понять как мне создать свой собственный ИИ и как спроектировать систему тестирования и обучения для моего ИИ.

## Deep RTS

### Обзор

В статье обозреваются среды для разработки стратегических ИИ. Все перечисленные среды используют только обучение с подкреплением в качестве основной технологии машинного обучения, так же как и Deep RTS. Кроме этого перечисленные среды работают значительно медленнее чем Deep RTS. Также были рассмотрены успехи машинного обучения и в игре Го, покере, нардах. В них применяли обучение с подкреплением и алгоритм дерева поиска Монте-Карло.

Среда Deep RTS расчитана на работу с классическими RTS, где нужно строить базу, развивать экономику, создавать армию и атаковать противника.

Deep RTS имеет несколько стандартных сценариев и конфигурацию для настройки поведения ИИ. Настройки конфигурации влияют на действия ИИ в начальной стадии игры, причём они должны применять сразу же на старте, а не через определённый промежуток времени.

Среда Deep RTS управляет боевыми единицами, которые делятся на 2 типа: здания(такие как казарма, ратуша) и юниты(рабочие, воины). Поведение юнитов определяется конечным автоматом. После создания юнита, он находится в состоянии ожидания, и из этого состояния он может перейти в нужное игроку или ИИ состояние.

Deep RTS управляет боевыми единицами с помощью действий, множество которых разделено на 2 абстрактных уровня:

1. Первый уровень - это действия, которые непосредственно влияют на среду, например, щелчок правой кнопкой мыши, щелчок левой кнопкой мыши, перемещение влево или выбор юнита.
2. Второй уровень абстракции - это действия, объединяющие действия предыдущего уровня, обычно это select-unit → right-click → right-click → move-left. Преимущество этой абстракции состоит в том, что алгоритмы могут фокусироваться на определенных областях внутри игрового состояния и позволяют создавать иерархические модели, каждая из которых специализируется на определённом виде задач(постройка новой базы, оборона, нападение).

### Аннотация

Обучение с подкреплением эта область которая достигла больших успехов за последнее время в моделировании оппонентов в компьютерных играх. Этот успех в первую очередь обусловлен обширными возможностями сверточных нейронных сетей, которые могут извлекать полезные данные из зашумленных и сложных данных. Игры являются отличными инструментами для тестирования и расширения границ новых алгоритмов RL(reinforcment learning, обучение с подкреплением), поскольку они дают ценную информацию о том, насколько хорошо алгоритм может работать в изолированных средах без реальных последствий. Стратегии в реальном времени (RTS) - это жанр, который имеет огромную сложность и бросает вызов игроку в краткосрочном и долгосрочном планировании. Существует много исследований, посвященных применению RL в RTS, и поэтому в ближайшем будущем ожидаются новые достижения. Однако на сегодняшний день существует несколько сред для тестирования ИИ в RTS. Среды в литературе часто бывают либо слишком упрощенными, такими как microRTS, либо сложными и не имеют возможности для ускоренного обучения на потребительском ПО таким как StarCraft II. Эта статья знакомит с игровой средой Deep RTS для тестирования передовых алгоритмов искусственного интеллекта для игр RTS. Deep RTS - это высокопроизводительная RTS-игра, созданная специально для исследований в области искусственного интеллекта. Он поддерживает ускоренное обучение, что означает, что он может учиться в 50 000 раз быстрее по сравнению с существующими RTS играми.

Deep RTS имеет гибкую конфигурацию, позволяющую проводить исследования в нескольких различных RTS-сценариях, включая частично наблюдаемые пространства состояний и сложность карт. Мы показываем, что Deep RTS оправдывает наши обещания, сравнивая его производительность с microRTS, ELF и StarCraft II на высокопроизводительном потребительском оборудовании. Используя Deep RTS, мы показываем, что агент Deep Q-Network выигрывает у агентов со случайным поведением в более 70% случаев. Deep RTS общедоступен по адресу https://github.com/cair/DeepRTS.

### Введение

Несмотря на многие достижения в искусственном интеллекте (AI) для игр, никакой универсальный алгоритм обучения с подкреплением(RL) не может быть применен к сложной окружающей среде игры без обширного манипулирования данными или настройки. Это включает традиционные стратегические игры В реальном времени (RTS), такие как WarCraft III, StarCraft II и Age of Empires. RL был недавно применен к более простой окружающей среде игры, такой как найденные в Аркадной среде обучения Галереи [1] (ALE) и настольные игры [2], но не был успешно применен к более продвинутым играм. Далее, существующая окружающая среда игры, которая предназначается для исследования AI, или чрезмерно упрощенна, такие как ALE или комплекс, такой как StarCraft II.

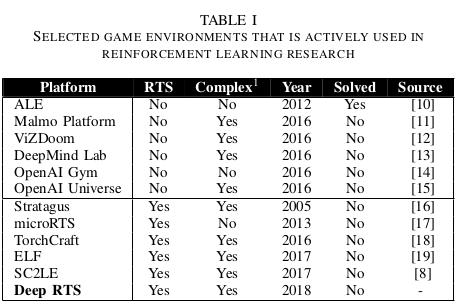
У RL в последние годы был огромный прогресс изучения, как управлять агентами от высоко-размерных сенсорных исходных данных как изображения. В простой окружающей среде это, как доказывали, работало хорошо [3], но является все еще проблемой для сложной окружающей среды с большим состоянием и местами действия [4]. Различие между простыми и сложными задачами в RL часто находится в том, как легкий это должно проектировать премиальную модель, которая поощряет алгоритм улучшать свою политику, не заканчиваясь в местном optima [5]. Для простых задач премиальная функция может быть описана только несколькими параметрами, в то время как в более требовательных задачах, алгоритм изо всех сил пытается определить то, чего премиальный сигнал пытается достигнуть [6]. Поэтому премиальная функция часто находится в литературе постоянная или однозначная переменная для большинства временных шагов, где только заключительный временной шаг определяет отрицательное или положительное вознаграждение [7] – [9]. В данной статье мы вводим Deep RTS, новая окружающая среда игры предназначалась для исследования глубокого обучения с подкреплением (DRL). Deep RTS - симулятор RTS, вдохновленный известной видеоигрой StarCraft II Blizzard Entertainment.

Данная статья структурирована следующим образом. Во-первых, Раздел II и Раздел III полностью обрисовывают в общих чертах предыдущую работу и центральные успехи, используя окружающую среду игры для исследования RL. Затем, Раздел IV вводит Глубокую окружающую среду игры RTS. Раздел V представляет Глубокую работу RTS, сравнение между известной окружающей средой игры и Глубоким RTS и результатами эксперимента, используя Глубокую Q-сеть в качестве агента в Глубоком RTS. Впоследствии, Раздел VI завершает вклад данной статьи и обрисовывает в общих чертах дорожную карту для будущей работы.

### Ранее созданные среды

Существует несколько захватывающих игровых сред в литературе, которые сосредотачиваются на исследованиях современных ИИ-алгоритмов. Небольшое количество игровых сред предназначены для жанра RTS. Один причина может состоять в том, что такие среды сложны по своей природе, и есть немного способов согласовать результаты с приемами предварительной обработки. Однако, важно включать RTS как часть активного исследования алгоритмов глубокого обучения с подкреплением, поскольку они позволяют строить долго-срочные планы. Этот раздел обрисовывает в общих чертах полный литературный обзор существующих платформ игры и окружающей среды и получен в итоге в Таблице I.

### A. Stratagus

 Stratagus - общедоступный игровой движок, который может использоваться, чтобы создать RTS игры. Wargus, клон Warcraft II, и Stargus, клон StarCraft - примеры игр, сделанных на игровом движке Stratagus. Stratagus не движок, который предназначается для машинного обучения явно, но несколько исследователей выполнили эксперименты в доказательной аргументации [20], [21] и q-изучение [22] использование Wargus. Stratagus все еще активно обновлен вкладами от сообщества.

### B. Arcade Learning Environment

Bellemare и др. представил в 2012 Arcade Learning Environment(ALE), который позволил исследователям провести cовременное исследование в общем глубоком изучении [10]. Пакет предоставил сотням Atari 2 600 окружающей среды, которая в 2013 позволила Мину и др. cделать прорыв, используя глубокое Q-изучение и A3C. Платформа была критически важным компонентом в нескольких достижениях в исследовании RL. [1], [3], [23]

### C. microRTS

microRTS - простая RTS игра, разработанная, чтобы провести исследование AI. Идея microRTS состоит в том, чтобы снять вычислительную тяжелую логику игры, чтобы увеличить работу и позволить исследователям проверить теоретические понятия быстро [17]. Логика игры microRTS детерминирована, и включена возможность для полных и частично заметных пространств состояний. Основная область исследования в microRTS - методы поиска дерева игры, такие как изменения поиска дерева Монте-Карло и минимакса [17], [24], [25].

### D. TorchCraft

В 2016 исследовательская группа развивала TorchCraft, проект, который позволяет сообществу изучать машинное обучение в StarCraft и TorchCraft, где только часть пространства состояний доступна [18]. В литературе TorchCraft использовался для глубокого исследования изучения [26], [27]. Есть также набор данных, который обеспечивает данные из-за 65 000 переигровок StarCraft [28].

### E. Malmo Platform

Проект Malmo - платформа, построенная на популярной игры Minecraft. Эта игра установлена в 3D окружающей среде, где цель состоит в том, чтобы выжить в мире опасностей. Статья платформы Malmo для «Artificial Intelligence Experimentation» Джонсона и др. утверждает, что у платформы есть все особенности, квалифицирующие его, чтобы быть платформой для общего исследования искусственного интеллекта. [11]

### F. ViZDoom

ViZDoom - платформа для исследования в визуальном обучении с подкреплением. В статье ViZDoom: Основанная на Doom Платформа Исследования AI для Визуального обучении с подкреплением Kempka и др., показала, что агент RL мог успешно учиться играть в Doom, игру шутера от первого лица, с поведением, подобным людям. [29]

### G. DeepMind Lab

В статье DeepMind Lab Битти и др. выпустила платформу для 3D навигации и задач решения головоломок. Основная цель DeepMind Lab состоит в том, чтобы действовать как платформа для исследования DRL. [13]

### H. OpenAI Gym

В 2016 Брокмен и др. от OpenAI выпустил GYM который они называли” набором инструментов для развития и сравнения алгоритмов обучение с покреплением”. GYM обеспечивает различные типы окружающей среды от следующих технологий: Алгоритмические задачи, Atari 2600, настольные игры, двигатель физики Box2d, двигатель физики MuJoCo и основанная на тексте окружающая среда. OpenAI также принимает веб-сайт, куда исследователи могут представить свою работу для сравнения между алгоритмами. GYM - открытый источник и поощряет исследователей добавлять поддержку своей среды. [14]

### I. OpenAI Universe

OpenAI недавно выпустил новую платформу изучения под названием Universe. Эта среда далее добавляет, что поддерживает среды, работающие в VNC. Это также поддерживает флеш-игры и приложения браузера. Однако несмотря на общедоступную политику OpenAI, они не позволяют исследователям добавлять новую окружающую среду к хранилищу. Это ограничивает возможности управления любой средой. Universe OpenAI - однако, важная для изучения платформа, поскольку у нее также есть поддержка настольных игр как Grand Theft Auto IV, которые допускают исследование в автономном вождении [30].

### J. ELF

Платформа исследования Extensive Lightweight Flexible (ELF) недавно присутствовала в NIPS с бумагой ELF: Обширная, Легкая и Гибкая Платформа Исследования для Стратегических игр В реальном времени. Эта работа фокусируется на исследовании RTS игр и является первой платформой, официально предназначающейся для этих типов игр. [19]

### K. StarCraft II Learning Environment

SC2LE (среда обучения StarCraft II) является оберткой API, которая облегчает доступ к состояниям игры StarCraft II с использованием языков, таких как Python. Цель состоит в том, чтобы позволить обучению с подкреплнием и алгоритмам машинного обучения использоваться в качестве AI для игроков игры. StarCraft II - сложная окружающая среда, которая требует короткого и перспективного планирования. Это трудно наблюдать корреляцию между действиями и вознаграждает из-за несовершенной государственной информации и задержанных вознаграждений, делая StarCraft II одной из самых трудных проблем до сих пор в исследовании AI.

### Обучение с подкреплением в играх

Хотя есть различные игровые среды с открытым исходным кодом подходящие для обучения с подкреплением обучения, немногие из них являются частью история успеха. Одной из причин этого является то, что нынешнее состояние современных алгоритмы кажутся нестабильными [30] и имеют трудности при определении оптимальной политики в окружающей среде с награждением множества целей [31]. В этом разделе представлены наиболее значимые достижения с использованием обучения подкрепления в игры.

## A. TD-Gammon

TD-Gammon - алгоритм, способный к достижению опытного уровня игры в Трик-траке настольной игры [7], [32]. Алгоритм был разработан Джеральдом Тесоро в 1992 в Научно-исследовательском центре Томаса Дж. Уотсона IBM. TD-Gammon состоит из искусственной нейронной сети (ANN) с тремя слоями и обучен, используя метод изучения укрепления под названием TD-лямбда. Лямбда TD-временный алгоритм изучения различия, изобретенный Ричардом С. Саттоном [33]. ANN повторяет по всем возможным шагам, игрок может выполнить и оценивает вознаграждение за то конкретное движение. Действие, которое приводит к самому высокому вознаграждению, тогда отобрано. TD-Gammon - первый алгоритм, который использует методы самоигры, чтобы улучшить параметры ANN.

## B. AlphaGO

В конце 2015 AlphaGO стал первым алгоритмом, который выиграет у профессионального игрока в Go. AlphaGO фреймворк обучения с подкреплением, которая использует дерево поиска Монте-Карло и две глубоких нейронных сети стоимости и стратегическую оценку [9]. Стоимость относится к ожидаемому будущему вознаграждению от состояния, предполагающего, что агент играет отлично. Стратегическая сеть пытается учиться, какое действие является лучшим в любой данной конфигурации правления. Самые ранние версии AlphaGO использовали данные тренировки от предыдущих игр, в которые играет человеческий profes-sionals. В новой версии, Ноле AlphaGO, только само - игра используется, чтобы обучить AI [34]. В недавнем обновлении AlphaGO был обобщен, чтобы работать на Chess и Shogi (японские Шахматы) только использование 24 часов, чтобы достигнуть сверхчеловеческого уровня игры [2].

## C. DeepStack

DeepStack - алгоритм, который может играть как эксперт в покере Техас Холдем. Этот алгоритм использует дерево поиска вместе с нейронными сетями, чтобы делать разумные действия в игре [35]. DeepStack — главный алгоритм, который нацелен на решение проблемы с несовершенной информацией. Алгоритм DeepStack доступен по ссылке <https://github.com/lifrordi/DeepStack-Leduc>.

## D. Dota 2

DOTA 2 - сложная конкурентная игра, где игрок управляет героем. Цель игры состоит в том, чтобы победить вражеских героев и разрушить их базу. В августе 2017 OpenAI создал AI, который победил профессиональных игроков в одном против игр. Обучение было сделано, только используя самоигру, и алгоритм изучил, как эксплуатировать механику игры, чтобы выступить хорошо в окружающей среде. DOTA 2 используется активно в исследовании, где следующая цель состоит в том, чтобы обучить AI играть в основанной на командной игре окружающей среде.

### Среда Deep RTS

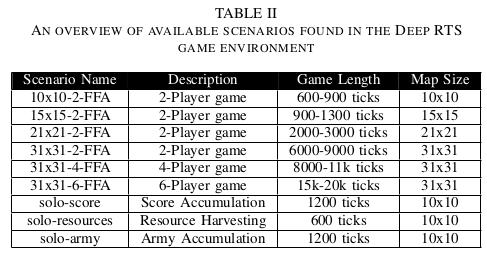
Необходимы новая игровые RTS среды нацеленные на изучение обучения с подкреплением. Немногие игровые среды имеют сложность, подходящую для современных исследований, и им не хватает гибкости существующих решений.

Игровая среда Deep RTS позволяет проводить исследования на разных уровнях сложности в планировании, реагирования и контроле. Вдохновением для этой статьи стали microRTS и StarCraft II, цель которой - создать среду, в которой будут решены проблемы обоих сред. Простейшие конфигурации Deep RTS являются детерминированными и недолговечными. Действия в недолговечной конфигурации непосредственно применяются к среде в течение следующих нескольких игровых кадров. Это делает корреляцию между действием и вознаграждением более наблюдаемой. Длительная конфигурация значительно усложняет пространство состояний, потому что тогда она становится временной проблемой, которая требует долгосрочного планирования. Deep RTS поддерживает абстракцию OpenAI Gym через Python API и является многообещающим инструментом для изучения подкрепления обучения.

### A. Цель игры

Задача Deep RTS состоит в том, чтобы построить базу, состоящую из ратуши, а затем стремиться расширить базу, используя собранные ресурсы для получения армии. Воинские части используются для проведения атак, где основной целью является уничтожение базы противника. Игроки начинают с одним рабочим юнитом. Основная задача рабочих юнитов - расширять базу, обороняться и собирать природные ресурсы, найденные во всем игровом мире. Здания могут далее порождать дополнительных юнитов, которые усиливают наступательные способности игрока. Чтобы игрок достиг конечного состояния, все подразделения противника должны быть уничтожены.

Обычные RTS игры состоит из трёх этапов: начальная стадия, средняя стадия и поздняя стадия. Ранняя игра - это этап сбора ресурсов и расширения базы. На средней стадии основное внимание уделяется военному и экономическому превосходству, в то время как во время поздней стадии игры обычно происходят главные сражения между игроками, пока игра не закончится.

 Поскольку Deep RTS предназначается для различных задач обучения с подкреплением, существуют игровые сценарии, такие как задачи сбора ресурсов, военные задачи и оборонительные задачи, которые сужают сложность полноценной игры RTS. В таблице 2 показаны девять сценариев, которые в настоящее время реализованы в игровой среде Deep RTS. Первые шесть сценариев - это обычные RTS-сценарии с возможностью участия 6 активных игроков в режиме каждый сам за себя. Сценарий «счёт для одного» имеет цель набрать как можно больше баллов в кратчайшие сроки. «Сбор ресурсов в одиночку» - это игровой режим, который фокусируется на сборе ресурсов. Агент должен найти баланс между расширением базы и сбором ресурсов, чтобы оптимально собрать как можно больше ресурсов. только-армия - это сценарий, в котором основной целью является быстрое наращивание армии и атака противника. Игровая среда Deep RTS позволяет исследователям создавать собственные сценарии с помощью гибкого интерфейса конфигурации.

### B. Игровая механика

Механика игры Deep RTS является гибкой и может быть скорректирована до начала игры. Таблица III показывает список доступных конфигураций. Важный выбор дизайна - позволить действиям воздействовать на окружающую среду без какой-либо временной задержки. Все действия привязаны к тиковому таймеру, который по умолчанию равен 10. Юниту требуется 10 тиков, чтобы переместить одну плитку, 10 тиков для одной атаки, и 300 тиков, чтобы построить здание. Тиковый таймер также включает в себя множитель, который позволяет регулировать количество тиков, равное секунде. Для каждой итерации игрового цикла счетчик тиков увеличивается и оцениваются таймеры тиков. Используя таймеры, игровое состояние напоминает то, как функционирует игровая механика StarCraft II, в то время как снижение значения таймера лучше напоминает microRTS.

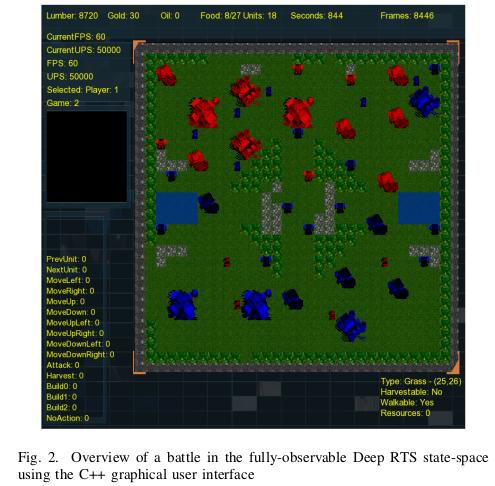
Все игровые сущности(юниты и здания) имеют конечный автомат который определяет текущее состояние. Рисунок 1 иллюстрирует часть логики, которая оценивается через конечный автомат. Объекты запускаются в состоянии порождения, переходя в состояние ожидания, когда процесс порождения сущности завершен. Состояние ожидания можно считать состоянием по умолчанию для всех сущностей, и оно изменяется только тогда, когда игрок взаимодействует с сущностью. Эта реализация позволяет исследователям модифицировать переходы состояний для создания альтернативной игровой логики.

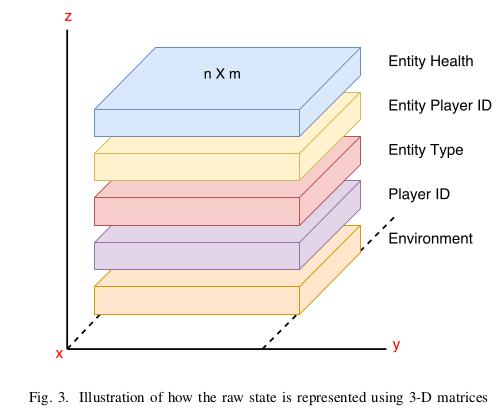
В таблице IV показаны доступные ресурсы и ограничения юнитов в игровой среде Deep RTS. В первую очередь есть три ресурса: золото, лес и нефть, которые рабочие могут добывать. Диапазон значений практически ограничен количеством ресурсов, имеющихся на игровой карте. Ограничение максимального количества еды и юнитов гарантируют, что игрок не будет производить юнитов слишком много.

### C. Графика

Игровой движок Deep RTS имеет два режима графического интерфейса в дополнение к режиму без камеры, который используется по умолчанию.

Основной графический интерфейс опирается на Python, а второй реализован на C ++. Версия Python не является интерактивной и может отображать только необработанное состояние игры в виде изображения. При использовании программного рендеринга процесс захвата изображений значительно быстрее, потому что копирование между GPU и CPU происходит медленно. Реализация C ++, показанная на рисунке 2, полностью интерактивна, что позволяет играть игроку в Deep RTS. На рисунке 3 показано, как необработанное игровое состояние представляется в виде трехмерной матрицы в режиме без камеры. Методы глубокого обучения часто предпочитают необработанные данные о состоянии игры вместо представления изображений в качестве сенсорного ввода. Это потому-что необработанные данные часто более конкретны с четкими закономерностями.



**D. Определение пространства действий**

Пространство действий игровой среды Deep RTS разделено на два абстрактных уровня. Первый уровень - это действия, которые непосредственно влияют на среду, например, щелчок правой кнопкой мыши, щелчок левой кнопкой мыши, перемещение влево и выбор юнита. Следующий уровень абстракции - это действия, объединяющие действия предыдущего уровня, обычно это select-unit → right-click → right-click → move-left. Преимущество этой абстракции состоит в том, что алгоритмы могут фокусироваться на определенных областях внутри игрового состояния и позволяют создавать иерархические модели, каждая из которых специализируется на задачах (планирование). Изначально Deep RTS содержит 16 различных действий на первом уровне и 6 действий на последнем уровне абстракции, но добавить дополнительные действия несложно.

### E. Итоги

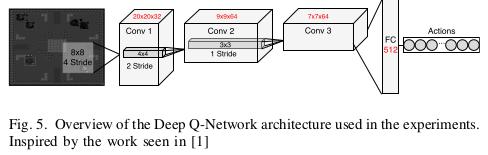
В этом разделе представлены некоторые гланые части, которые игровая среда Deep RTS предоставляет для изучения обучения с подкреплением. Он предназначен для точного измерения производительности алгоритмов, имеющих стандартизированный API через OpenAI Gym, который широко используется в сообществе использующее обучение с подкреплением.

### Эксперементы

**A. Производительность Deep RTS**

Цель Deep RTS состоит в том, чтобы точно моделировать RTS сценарии. Работа измеряется тем, как быстро движок игры обновляет состояние игры, и как быстро состояние игры может быть представлено как изображение. Некоторые эксперименты предполагают, что это выгодно, чтобы отдать графику игры на центральном процессоре вместо GPU. Поскольку у GPU есть отдельная память, есть серьезное узкое место, копируя буфер экрана от GPU до центрального процессора.

Рисунок 4a показывает корреляцию между частотой кадров и размером карты игры. Наблюдая данные, ясно, что у размера карты есть O (n) штраф работе частоты кадров. Жизненно важно сохранить эту линейность, и оптимально иметь постоянное исполнение O (1) за обновление игры. Рисунок 4 расширяет эту оценку, проверяя влияние, которое единица оказывает на производительность игры, составляя в среднем 1 000 игры для всех размеров карты. Данные указывают, что предприятия оказывают показательное влияние на работу частоты кадров. Причина этого - прежде всего, алгоритм поиска пункта скачка, используемый для новаторской единицы. Новаторский алгоритм может быть отключен, используя таможенные конфигурации.

 Глубокая окружающая среда игры RTS высокоэффективна с немногими элементами, которые значительно уменьшают работу частоты кадров. В то время как некоторая механика, а именно, новаторская, является значительной частью петли обновления, это может быть дезактивировано конфигурациями, чтобы оптимизировать работу далее.

### B. Сравнение Deep RTS с существующими средами

Есть существенные различия между работой в предназначенном исследовании игр и нацеленными к играм. Таблица V показывает, что различие частоты кадров колеблется от 60 до 7 000 000 для отобранной окружающей среды. Высокий framerate важен, потому что некоторые алгоритмы исследования часто требуют быстрой оценки будущих состояний через форвард-поиск. Таблица V показывает, что microRTS, ELF и Deep RTS лучше в работе по сравнению с другими средами. Deep RTS измерен, используя самую большую доступную карту (Таблица II), имеющая максимальное количество юнитов равное 20 на игрока. Это приводит к выполнению 24 000 обновлений в секунду. Движок Deep RTS может также отдать состояние игры максимум с 7 000 000 обновлениями в секунду, используя минимальную конфигурацию. Это - огромный прорыв по сравнению предыдущими работами и могло позволить алгоритмам с ограниченным по времени бюджетом сделать более глубокие поиски дерева.

### C. Использование Q-обучения в Deep RTS

На самом базовом уровне Q-Learning использует таблицу для хранения пар (s, a, r, s`), где s - это состояния, a - действия, r - награда и s`- следующее состояние. Вместо этого для нелинейной функции приближения можно использовать приближение нелинейной функции Q (s, a; θ). Это называется Deep-Q Learning. θ описывает настраиваемые параметры (веса) для функции приближения. Искусственные нейронные сети используются в качестве функции приближения для Q-таблицы, но цены не будут стабильны [3]. Использование искусственных нейронных сетей очень похоже на сжатие изображений JPEG. Сжатие с потерями, и некоторая информация теряется во время сжатия. Глубокое Q-Learning, таким образом, нестабильно, так как значения могут быть неправильно закодированы во время обучения [36].

В данной статье представлены экспериментальные результаты использования архитектуры Deep Q-Learning из [3], [37]. На рисунке 5 показана модель сети, а на рисунке 6 - усредненная потеря обучения на 100 агентов. Агент использует представления игровых состояний в градациях серого с дополнительным сверточным слоем для уменьшения времени обучения, но также может достигать сопоставимых результатов после приблизительно 800 эпизодов обучения с точной архитектурой из [3] 2. График показывает, что агент быстро изучает корреляцию между состоянием игры, действием и функцией вознаграждения. Потеря быстро стабилизируется при относительно низком значении, но вполне вероятно, что очень небольшая оптимизация параметров оказывает значительное влияние на производительность агента.

На рисунке 7а показан коэффициент выигрыша против ИИ со случайной стратегией игры. Агент быстро узнает, как действовать лучше, чем при случайном поведении, и достигает 70% выигрыша в эпизоде 1 250. На рисунке 7b показан тот же агент, играющий против стратегии, основанной на правилах. График показывает, что Deep Q-Network может достичь среднего выигрыша в 50% за 1000 игр. Эта стратегия считается игроком среднего уровня легкой, где ее стратегия заключается в расширении базы по направлению к противнику и наращивании вооруженных сил примерно через 600 секунд. На рисунке 2 показано, как игрок на основе правил (синий) расширяет базу, чтобы получить преимущество.

Экспериментальные результаты, представленные в этой статье, показывают, что игровая среда Deep RTS может использоваться для проверки алгоритмов глубокого обучения с подкреплением. Глубокая Q-сеть не достигает опыта профессионального игрока, но действует подобно игроку с легким или средним уровнем игры, что является хорошим шагом к высокоуровневому ИИ.

### 

### 

### Заключение и будущие работы

Данная статья - вклад в продолжение исследования глубокого обучения с подкреплением в играх жанра RTS. Статья подводит итоги по предыдущим работам и обрисовывает в общих чертах некоторые, но существенные истории успеха в изучении обучения с подкреплением. Среда Deep RTS высокоэффективный симулятор RTS, который позволяет быстро исследовать и тестировать новые методы обучения с подкреплением. Это успешно заполняет промежуток между жизненным симулятором игры microRTS и StarCraft II, который является конечной целью для исследования изучения обучения с подкреплением в играх жанра RTS.

Надежда состоит в том, что Deep RTS может принести проницательные результаты в сложных проблемах RTS [17] и что это может быть полезным инструментом в будущих исследованиях.

Хотя Deep RTS готова к использованию, есть несколько улучшений которые могут быть применены к окружающей среде.

Следующие пункты отмечены для внедрения в следующие версии Deep RTS:

* Добавить возможность разработчикам LUA использовать Deep RTS.
* Реализовать универсальный интерфейс для пользовательской визуализации графики.
* Осуществите дуплекс WebSockets и ZeroMQ, чтобы позволить любому языку взаимодействовать с Deep RTS
* Осуществите альтернативные новаторские алгоритмы, чтобы увеличить работу для некоторых сценариев
* Добавьте возможность использовать туман войны

### Благодарность

Мы хотели бы поблагодарить Сантьяго Онтанона из университета Drexel за его превосходную работу над microRTS. microRTS, является для нас ценным инструментом для критериев в области обучения с подкреплением и продолжает быть переходной средой для исследования алгоритмов поиска дерева.

# Список литературы

1. Javier Villanueva Forner «USE OF MACHINE LEARNING TECHNIQUES IN VIDEOGAMES»;dissertation is submitted for the bachelor’s degree of Video Game Design and Development [Электронный ресурс]// URL: https://arxiv.org/abs/1611.00625
2. Per-Arne Andersen, Morten Goodwin, Ole-Christoffer Granmo «Deep RTS: A Game Environment for Deep Reinforcement Learning in Real-Time Strategy Games» [Электронный ресурс]// URL: http://repositori.uji.es/xmlui/bitstream/handle/10234/175613/MEMORIA\_VillanuevaFornerJavier.pdf?sequence=1&isAllowed=y
3. Kun Shao, Yuanheng Zhu, Member, IEEE and Dongbin Zhao, Senior Member «StarCraft Micromanagement with Reinforcement Learning and Curriculum Transfer Learning» [Электронный ресурс]// URL: https://arxiv.org/abs/1808.05032
4. Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, Nicolas Usunier «TorchCraft: a Library for Machine Learning Research on Real-Time Strategy Games» [Электронный ресурс]// URL: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/
5. Бобков А. C., Егошин А. В. «ОБЗОР СПОСОБОВ ОПТИМИЗАЦИИ ГРАФА ДОРОЖНОЙ СЕТИ В ЗАДАЧЕ МАРШРУТИЗАЦИИ». [Электронный ресурс]// URL:[https://science.volgatech.net/upload/documents/conf/np\_tm/%D1%87%D0%B0%D1%81%D1%82%D1%8C%203.pdf](https://science.volgatech.net/upload/documents/conf/np_tm/часть 3.pdf)
6. Hendrik Baier, Peter I. Cowling «Evolutionary MCTS for Multi-Action Adversarial Games»[Электронный ресурс]// URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwjqpb-r-qbgAhWI8ywKHdj5B5YQFjAAegQICRAC&url=https%3A%2F%2Fhendrikbaier.jimdo.com%2Fapp%2Fdownload%2F13604570727%2Fevolutionary%2520mcts%2520for%2520multi-action%2520adversarial%2520games.pdf%3Ft%3D1532703212&usg=AOvVaw3M9xX0AbJl4h5uYAy8GBBh