

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Дальневосточный федеральный университет»**

**(ДВФУ)**

|  |
| --- |
| **ШКОЛА ЕСТЕСТВЕННЫХ НАУК**  **Кафедра информатики, математического и компьютерного**  **моделирования**  **ОТЧЕТ**  **Запуск интерактивной задачи с помощью Spawner**  Студент группы М8103  Костин Дмитрий Александрович  г. Владивосток  2017 |
|  |
|  |
|  |

Содержание

[Содержание 2](#_Toc471704901)

[1. Описание предметной области 3](#_Toc471704902)

[2. Описание протокола 5](#_Toc471704903)

[3. Постановка задачи 7](#_Toc471704904)

[4. Этапы выполнения работы 7](#_Toc471704905)

[4.1. Компиляция и запуск 7](#_Toc471704906)

[4.2. Изменения кода движка 9](#_Toc471704907)

[4.3. Запуск с помощью механизма многоагентных задач в Spawner 10](#_Toc471704908)

[5. Заключение 16](#_Toc471704909)

[Список литературы 17](#_Toc471704910)

## Описание предметной области

Возможность запуска интерактивных и многоагентных задач была реализована в модуле Spawner студентом Ротановым Д.В. в 2015 году [[1]](#_Список_литературы).

Потоки данных стандартной задачи, выполняемой в процессе проверки решения в системе соревнований CATS, могут быть представлены в виде следующей схемы (рис. 1):

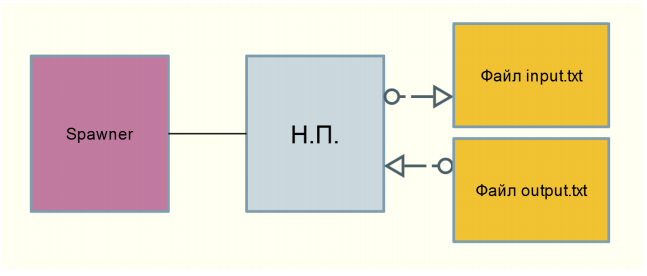


Рис. 1: Потоки данных в стандартной задаче

Где Н.П. – нормальная программа (решение стандартной или интерактивной задачи, либо интеллектуальный агент). Последовательность действия для проверки стандартной задачи включает в себя взаимодействие тестирующей системы и решения задачи. Входные и выходные данные чаще всего передаются через файлы, указанные в условиях задачи или через стандартные потоки STDIN, STDOUT.

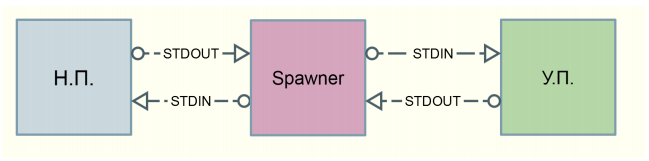


Рис. 2: Потоки данных в интерактивной задаче

При использовании игровой стратегии, проверка решения рассматрива- ется как игра двух противников — тестирующей системы (интерактор) и тести- руемой программы (решение участника). Первый ход осуществляет тестиру- ющая система (проверяющая программа). Далее ход переходит к тестируемой программе. Она запускается на входных данных, подготовленных тестирующей системой, и выдает некоторый результат. Ход вновь получает тестирующая система. Она анализирует результат, полученный тестируемой программой. На основе результатов анализа принимается решение о продолжении тестирования. Если тестирование продолжается, то проверяющая система подготавливает новый набор входных данных, при этом может быть использована информация , полученная от тестируемой программы, на предыдущих шагах. После чего ход снова передается тестируемой программе, и так далее (рис. 2).

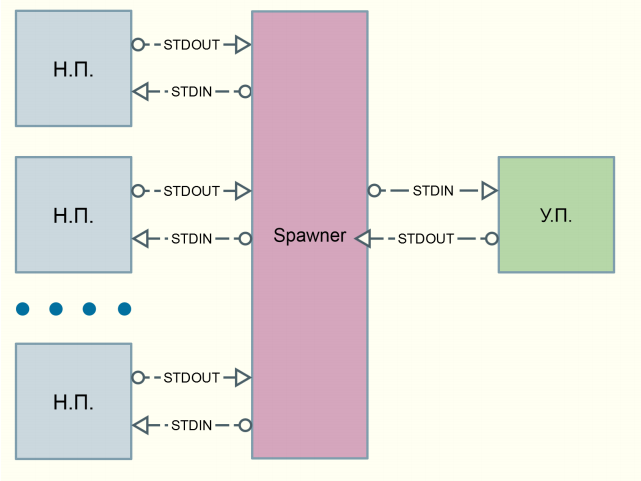


Рис. 3: Потоки данных в многоагентной задаче

В случае многоагентных задач увеличивается количество одновременно взаимодействующих программ и у тестирующей программы (контроллера) по- является необходимость управлять исполнением нормальных программ через Spawner. При этом взаимодействие с контроллером по прежнему осуществля- ется через два стандартных потока ввода-вывода, что наталкивает на мысль о необходимости мультиплексирования каналов связи по некоторому протоколу (рис 3).

## Описание протокола

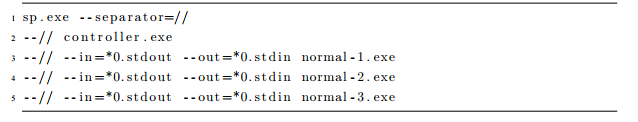
Для запуска интерактивных задач в модуле Spawner предусмотрен интерфейс запуска нескольких процессов.

Необходимо указать последовательность символов — разделитель, кото- рая будет отделять секции аргументов командной строки для каждого за- пускаемого процесса. Это осуществляется с помощью ключа --separator.



Таким образом будет запущено 4 процесса. Для настройки параметров процесса нужные флаги должны помещаться в соответствующую секцию.

Для того, чтобы осуществить связь потоков ввода вывода необходимо ис- пользовать флаги –in=<name> для STDIN и –out=<name> для STDOUT, где <name> имеет формат \*<число>.stdin или \*<число>.stdout, что позволяет сослаться на соответствующие потоки запускаемого процесса с индексом . Процессы нумеруются в порядке их передачи Spawner, начиная с 0. Так же может быть указано значение –in=std или –out=std, что осуществляет перенаправление на стандартный ввод/вывод. Следующая командная строка связывает STDOUT контроллера с STDIN каждого нормала и STDIN контроллера с STDOUT каждого нормала:



Для того чтобы пометить запускаемый процесс как контроллер требуется передать ключ –controller:



Контроллер и нормал запускаются с помощью Spawner, передаваемые в качестве аргументов командной строки. Предъявляются следующие требова- ния:

– контроллер должен быть помечен флагом –controller.

– контроллер должен быть единственным

– наличие контроллера должно переводить spawner в режим управления

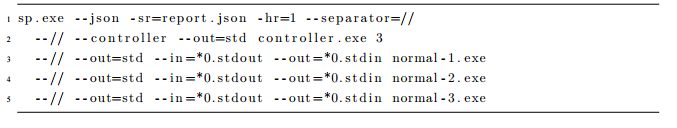
– остальные требования применимы только в режиме управления

– Количество нормалов должно быть передано контроллеру первым аргу- ментом командной строки

– stdin каждого нормала должен быть соединён с stdout контроллера с по- мощью –in=\*0.stdout

– stdout каждого нормала должен быть соединён с stdin контроллера с по- мощью –out=\*0.stdin

Пример командной строки запуска:



Для общения контроллера и нормала используется механизм сообщений.

Сообщения имеют форму <заголовок><тело><перевод строки>. Тело мо- жет состоять из любых символов кроме символов перевода строки или возвра- та каретки. Заголовок должен всегда кончаться символом #. Перед символом # может быть латинская буква, перед которой в свою очередь может быть целое число.

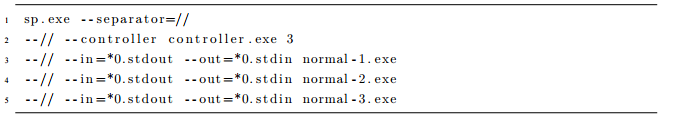
– контроллер и нормал должны отсылать сообщения через stdout

– stdout должен быть сброшен (flushed) после отсылки сообщения

– сообщения должны заканчиваться символом перевода строки

При отсылке сообщений нормалу, контроллер обязан указать в заголовке сообщения индекс нормала, которому предназначено сообщение. Нормалы индексируются в порядке их передачи Spawner в качестве аргументов командной строки, начиная с единицы.

Например для такой схемы запуска:



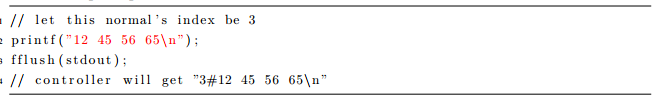
Чтобы отослать сообщение ”Foo”приложению normal-1.exe контроллер должен выполнить следующий код:



Контроллер может сообщить Spawner об ожидании им ответа от конкрет- ного нормала: <i>W#<перевод строки>.

Контроллер может запросить Spawner прекратить выполнение процесса указанного нормала с помощью сообщения <i>S#*.* Spawner не должен сообщать об ошибке на запрос остановки, если индекс неверен.

Spawner обязан дополнить слева сообщения normal его индексом и сим- волом # чтобы образовать заголовок сообщения, который будет разобран кон- троллером.



## Постановка задачи

Необходимо провести запуск интерактивной задачи, представленной игровым движком игры Ultimate Tic-Tac-Toe с соревновательной площадки The AI Games [[2]](#_7._Список_литературы) и парой ботов, с помощью модуля Spawner.

## Этапы выполнения работы

В рамках данной работы были выполнены следующие задачи:

1. Компиляция и запуск движка игры с парой соревнующихся ботов с помощью Spawner, а также последующая визуализация результатов
2. Изменение части движка игры для использования в качестве контроллера в многоагентной задаче
3. Запуск движка игры и пары ботов с помощью механизма многоагентных задач в Spawner

## Компиляция и запуск

Для компиляции и запуска движка игры с парой соревнующихся ботов был написан скрипт sp\_simple.sh, в котором Spawner был использован с ключом --separator:

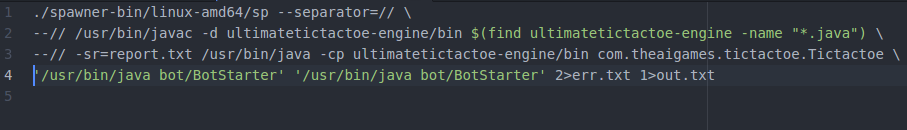


Рис. 3: Код скрипта sp\_simple.sh

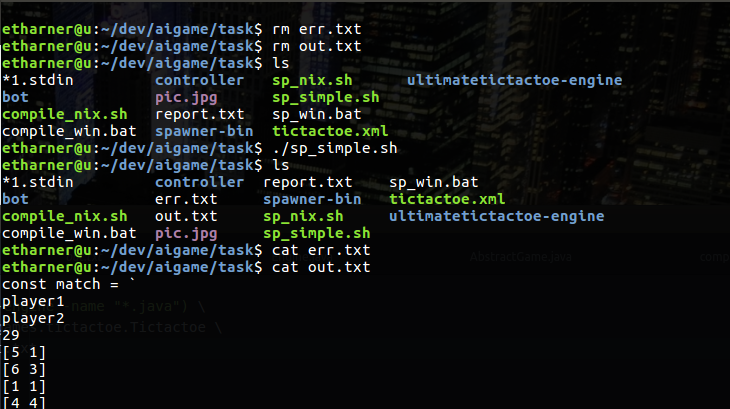


Рис. 4: Запуск и проверка скрипта sp\_simple.sh

При этом имеется возможность передачи любых ботов в скрипт, а также любых доступных параметров spawner, вроде ограничения по времени, памяти и тд.

## Изменения кода движка

Рис. 5: Визуализация результатов выполнения скрипта sp\_simple.sh

Для адаптации движка игры к запуску в среде многоагентной задачи в качестве контроллера были произведены следующие изменения в коде:

1. Количество входных аргументов программы уменьшено до одного, обозначающего количество нормалов в задаче (до этого два аргумента, передаваемые в программу, обозначали имена процессов, в данном случае пару ботов)
2. Произведена отвязка потоков ввода-вывода от процессов, в программе используется общий стандартный поток ввода-вывода
3. Изменена структура посылаемых сообщений (в соответствии с главой [2](#_Описание_протокола) к сообщению слева добавляется заголовок, хранящий в себе номер нормала, к которому посылается сообщение, а само сообщение передается по стандартному потоку вывода)
4. Объекты, хранящие информацию об игроках и соответствующих им процессах теперь хранят номер нормала, означающий процесс игрока
5. Добавлен разбор входящих сообщений от нормалов (в соответствии главой [2](#_Описание_протокола) перед передачей сообщения от нормала в логику игры из сообщения извлекается номер нормала)
6. Обработка стандартного потока ввода теперь обрабатывается в том же потоке, в котором запущен сам движок
7. После каждой отсылки сообщения нормалу добавляется сообщение ожидания ответа
8. После завершения матча нормалам отправляется сообщение, завершающее их работу

Подробные изменения в коде, а так же скрипты и визуализатор возможно посмотреть с помощью системы контроля версий на сайте <http://github.com> [3].

## Запуск с помощью механизма многоагентных задач в Spawner

На данный момент возможность запуска многоагентных задач с помощью Spawner реализована только для операционных систем семейства Windows, поэтому ниже приведен скрипт для этой ОС.

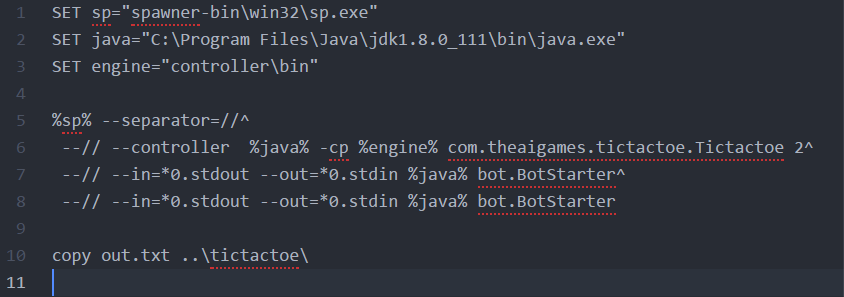


Рис. 6: Код скрипта sp\_win.bat

Ниже продемонстрирован отчет Spawner об успешно завершенной многоагентной задаче (примечание: полный отчет генерируется только с ключом --json -sr=<имя файла>.json, без ключа json в любом другом формате в отчет заносятся только данные о запуске последнего переданного нормала).

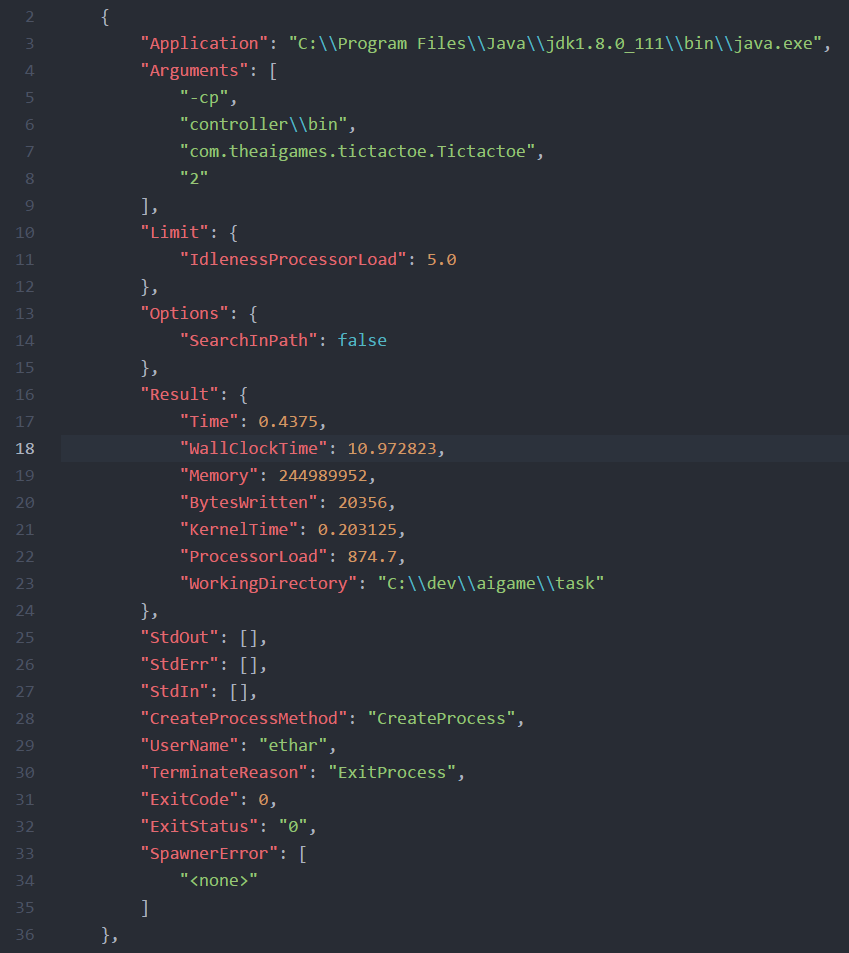


Рис. 7: Отчет Spawner о запуске контроллера

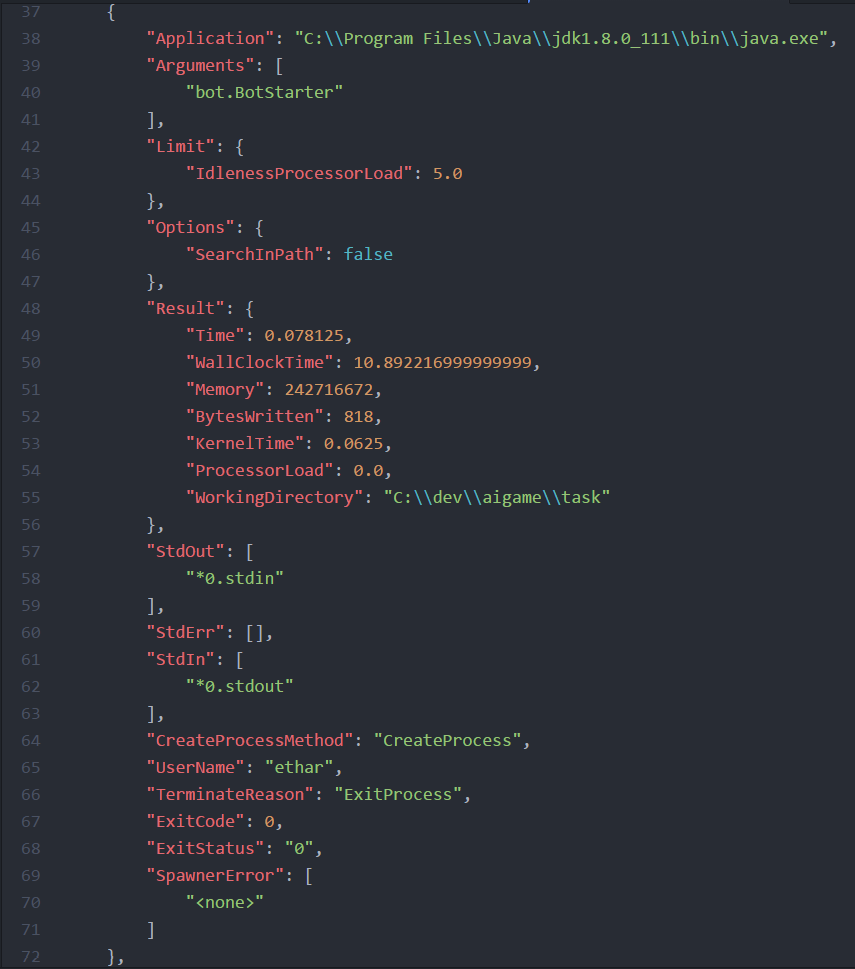


Рис. 8: Отчет Spawner о запуске первого нормала

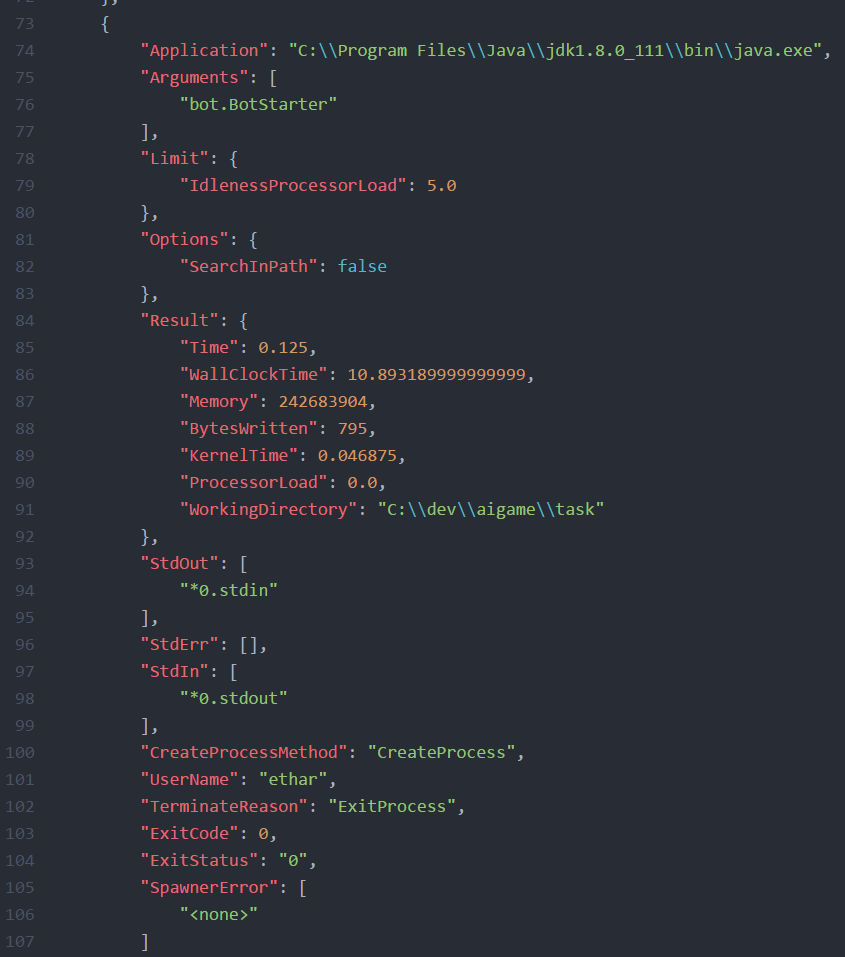


Рис. 9: Отчет Spawner о запуске второго нормала

При отладке также был указан ключ --out=std для Spawner для отображения общения контроллера и нормалов в консоли, часть вывода представлена на рис. 10.

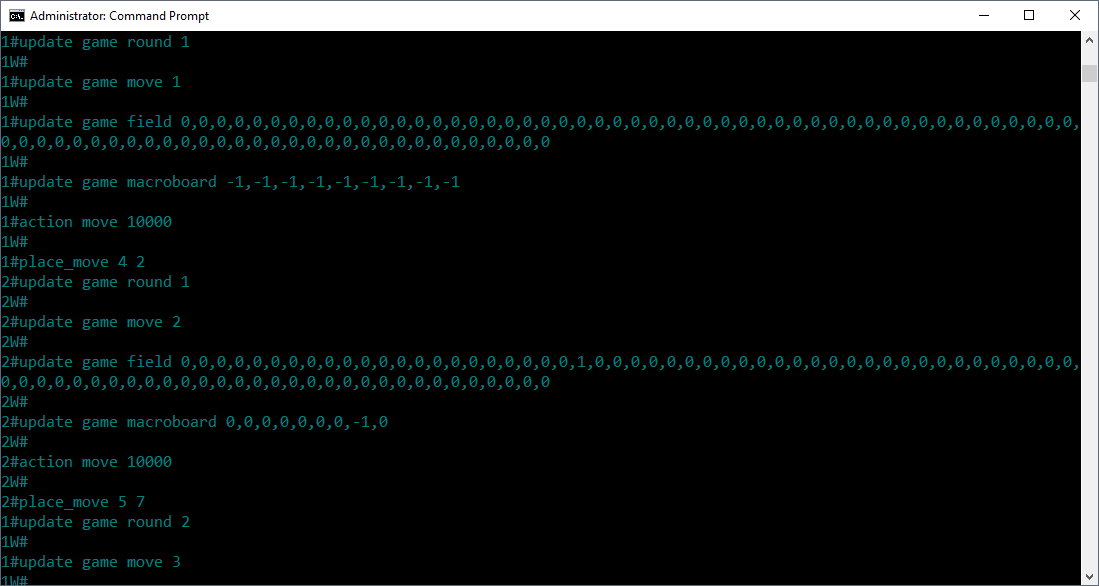
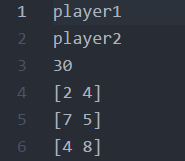
Результат матча записывается на стороне контроллера в отдельный файл out.txt, для тестового матча часть файла представлена на рис. 11.

Рис. 11: Результат матча: начало файла

Рис. 10: Общение контроллера и нормалов

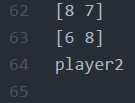
В конце файла указан победитель (рис. 12).

Рис. 12: Результат матча: конец файла

## Заключение

Таким образом, в результате выполнения работы был изучен протокол взаимодействия контроллера и нормалов в многоагентной задаче, написаны скрипты для запуска движка и ботов в совместном и многоагентном режиме и модифицирован код движка для работы с многоагентной задачей. Также были указаны особенности Spawner в области входных параметров и применимости в различных операционных системах.

## Список литературы

[1] – Ротанов Д.В., Кленин А.С. Дипломная работа «Контролируемое исполнение программ в многозадачном окружении». — ДВФУ, 2015г.

[2] – The AI Games [http://theaigames.com/](http://theaigames.com/%20)

[3] – Репозиторий aigame <https://github.com/etharner/aigame/commits/master>