УДК 004.514.62

***Дёмин В.В***

***Научный руководитель: Дунец А.П., доцент***

**Реализация модели робота игрока для Robocup Soccer Simulation Server**

**Введение**

RoboCup — международные соревнования среди роботов, основанные в 1993 году. Официальная цель проекта — к середине 21-го века команда полностью автономных человекоподобных роботов футболистов должна выиграть футбольный матч, соблюдая правила FIFA, у победителя Чемпионата мира.

Simulation league - это часть проекта Robocup, которая направлена на моделирование игры в простейшем 2D - варианте, без относительно реальных реализаций роботов-футболистов.

В лиге симуляции принимают участие две команды с одиннадцатью автономными программами (которые называют агенты), играющие в футбол на двухмерном виртуальном стадионе, который предоставляется центральным сервером под названием SoccerServer. На сервере содержится вся необходимая информация об игре, такая как позиции игроков и мячика, физика, направления игроков и другие параметры и свойства виртуального мира. Игра основана на сообщениях между сервером и агентом. С одной стороны каждый игрок получает данные с его виртуальных сенсоров (визуальные, акустические и физические), а с другой может формировать некоторые базовые команды (ускорение, поворот, удар) в целях оказания воздействий на окружающую среду.

При построении своей реализации агента-игрока были рассмотрены программные модули различных команд, занимавших в последние года места в топ-10 чемпионата мира Robocup.

**1. Структура агента-игрока**

Как и любой другой программный продукт, робот-агент должен содержать ряд модулей. Специфика Robocup SoccerServer заключается в том, что агент должен уметь буквально всё, от возможности отправлять и принимать сообщения по средствам UDP сокетов, алгоритмов управления поворотом направления обзора относительно расположения мячика на поле до алгоритмов коллективного поведения. Всё это приводит к явному разделению модуля агента на три уровня:

- Низкоуровневая логика — простейшие действия, такие как поворот агента на угол, отправка/принятие сообщения от сервера, служащие в первую очередь основой для более сложных действий. Этот уровень является абстракцией, драйвером скрывающим от программиста низкоуровневую реализацию и дающий возможность работать непосредственно с действиями робота-игрока, т. е. манипулированием им.
- Среднеуровневая логика — представляет собой наборы из действий низкоуровневой логики. Каждое действие среднеуровневой логики направлено на решение какой либо ситуации происходящей с роботом: оценка ситуации, исполнения какого либо действия (удар, отбор мяча, ловить мяч в воротах). Сами по себе эти действия уже составляют серьёзную базу, и на их основе возможно полноценное функционирование команды.
- Высокоуровневая логика — это алгоритмы коллективного поведения, искусственного поведения, а так же список предписаний для агентов, при различных ситуациях на поле, представляющий комплексный подход в решении данной задачи. Если вышеописанные уровни как правило совпадают или мало

различаются у большинства команд, то высокоуровневая логика является концептуальной задачей для каждой команды, т. к. это и есть реализация «мозгов» команды.

**2. Особенности Robocup Soccer Simulation Server**

Для того, что бы лучше понять для какой системы будет вестись разработка следует дать более подробное описание программного обеспечения, используемого в лиге симуцляции.

Robocup Soccer Simulator Server — это исследовательская и обучающая среда для мультиагентных систем и искусственного интеллекта. Среда позволяет двум командам с 11-ю программными автономными роботам играть в футбол. Сам сервер представляет собой сервер и набор утилит и имеет следующую структуру:

- Сервер — непосредственно сам сервер
- Мониторы — программы, служащие для визуализации происходящего на поле в реальном времени. Так же служат для отладки агентов-игроков
- Агенты-игроки — модули команд-соперников, предоставляющих по 11 виртуальных агентов с каждой стороны поля
- Тренеры — виртуальные агенты, с возможностью обозревать всё поле одновременно. Их месторасположение — бровка поля, как и у тренеров в настоящем футболе. Тренер является важной частью каждой команды и способствует более координированному её поведению.
- Лог игрока — программа предоставляет возможность вести лог всех действий на сервере, тем самым предоставляя возможность просмотреть ещё раз игру и более тщательно оттестировать модули агента-игрок.

Сервер это программа с помощью которой различные команды могут играть в футбол. На время игры он представляет собой клиент-серверную систему, где не имеет значения как сделаны программы-игроки. Единственное требование к их реализации это использование связи с сервером по средствам протокола UDP/IP. Это следует из того, что все сообщения между сервером и клиентом (агентом-игроком) исполнено через UDP сокеты. Каждый клиент представляет собой отдельный процесс и подсоединяется к серверу через специальный порт. После соединения с сервером, все сообщения между ними происходят только через этот порт. Каждая команда может иметь до 12 клиентов, 11 агентов-игроков (10 полевых и 1 вратарь) и тренера. Игроки посылают серверу сообщения с действиями, которые они хотят выполнить (ударить мяч, повернуться, бежать и т.д.). Сервер получая эти сообщения, выполняет запросы, и обновляет информацию о среде соответственно. Так же сервер обеспечивает всех игроков сенсорной информацией (визуальной — позиция объектов на поле или данные о физических свойствах игрока, таких как выносливость или скорость).

Сервер работает в режиме реального времени. Время делится на дискретные временные интервалы — циклы. Таким образом низкая производительность игроков, приводит к пропуску действий, что очень сильно влияет на результат команды в целом.

Последняя версия сервера, на дату написания данной статьи, является 14.0.3. Основной проблемой в исследовании сервера и написания основных модулей агента является недоработанная документация. В результате большинство команд начинает свои исследования с разбора и модернизирования исходных кодов уже готовых реализаций. И как правило их собственные реализации основываются на исходном коде низкоуровневой и среднеуровневой логики.

Монитор представляет собой средство визуализации которое позволяет наблюдать, что же происходит на поле внутри сервера в течении игры. В настоящий момент есть две реализации монитора rcssmonitor — написанной с использованием библиотеки Qt4 и rcssmonitor_classic, использующий для вывода графической информации на экран фрэимбуфер. Информация, предоставляемая на экране, содержит счет, названия команд, и

позиции всех игроков и мяча. Так же монитор содержит простейшие команды управления сервером. Например, когда обе команды присоединены, копка «Kick-Off» (вбросить мяч) в интерфейс монитора позволяет человеку-судье начать игру.

Основные особенности монитора:

- Возможно приближать различные участки поля. Это свойство обычно используют для отладки программ и поиска ошибок в реализации;
- Положение каждого игрока на поле и мяча может выводится в консоль, в каждый цикл сервера;
- Различного вида информация может быть отображена на мониторе, такая как угол обзора, выносливость, вид игрока;
- Игроки и мяч могут быть перемещены с помощью мышки.

На самом деле функционирование сервера не зависит от наличия подсоединенного монитора. В том случае, если он необходим, определенное количество мониторов соединяется с сервером одновременно.

Программу лог игрока можно представить как видео съёмку. Это утилита для проигрывания прошедших матчей. Когда запущен сервер, есть возможность задать ему опцию сохранения данных на жесткий диск. Когда лог игрока используется вместе с монитором, появляется возможность просматривать матч столько раз, сколько это необходимо. Это очень полезная утилита, позволяющая наглядно изучать стратегии соперников, их минусы и плюсы. А так же оценивать свои возможности, поиск дырок в оборонной тактике или недостатков в нападающей. Так же лог игрока представляет возможность каждой команде сохранять свою игру и далее легко вставлять в свои презентации.

## 3. Алгоритмы игрока

Для достижения быстрого результата и более глубокого ознакомления с аспектами программирования для сервера симуляции было решено взять открытый код чужой команды. На сегодняшний день есть несколько реализаций агентов-игроков, имеющих популярность у команд-новичков. Многие из них были проанализированы и был выбран вариант, наиболее подходящий составленной логической схеме работы модуля.

За основы низкоуровневых и среднеуровневых алгоритмов были взята реализация агента-игрока команды UvA Trilearn университета Амстердама. Она имеет ту же концепцию разноуровневой структуры организации модуля игрока, которая была принята в реализации данной работы.

### 3.1. Низкоуровневые методы

В данной статье нет возможности описать полностью все классы и методы, реализованные в ходе разработки и перенятые с других исходных кодов. Поэтому следует кратко описать основные реализованные функции.

За соединение с сервером отвечает класс Connection, в котором реализованы функции подсоединения, отсоединения, посылки информации серверу. Все данные передаются и принимаются посредству UDP сокетов. За получение необходимой информации из сообщений создан класс Parse, содержащий всевозможные методы для разделения сообщения на части.

Реализацей основных функций игрока и основных функций тренера являются классы BasicPlayer и BasicCoach соответственно. Здесь реализован весь функционал, который используется в более высокоуровневых методах.

Абстрактным классом для представления объектов на поле является класс Object. Он является базовым для всех объектов и содержит методы получения информации и задания данных объектам. Классы наследники Object это DynamicObject и FixedObject, где первый описывает объекты, которые могут перемещаться за момент времени по игровому полю, второй — статические объекты, такие как флаги на углах поля, лини разметки, ворота.

Несмотря на то, что эти два класса можно так же отнести и к среднеуровневой логике, их методы, в основном, представляют простейшие действия, используемые для построения более сложных комбинаций.

### 3.2. Среднеуровневые методы

На основе вышеописанных классы строятся более сложные классы. Среднеуровневые методы — это комбинации низкоуровневых методов, или более сложные действия.

Одним из таких классов является VecPosition. Он содержит x и y позиции, с помощью которых может рассчитывать расстояния до других объектов их направления и т. д. Все что связано с навигацией агента-игрока реализуется в этом методе.

Класс Stamina отвечает за своевременный отдых игрока, и не позволяет ему экономить силы во время игры.

Класс Time представляет реализует методы мониторинга времени сервера для своевременной реакции на задержки в работе модуля. Это производится так же за счёт класса Timing, реализующего внутренний таймер агента. Основной необходимостью данных классов и их взаимодействия является ограничение по времени одного цикла симуляции. Его длительность составляет 300мс. И если за это время от агентов не пришло ни одного сообщения или команды, то в следующей итерации ничего не произойдет и агенты останутся на своих местах. Что бы не допустить такой ситуации очень важно вовремя её отследить и начать использовать более простые алгоритмы, требующие меньше процессорного ресурса.

BallObject и PlayerObject представляют классы наследники DynamicObject и описывают объект мяча и объект игрока сообтветственно.

### 3.3. Высокоуровневые методы

Высокоуровневые методы сами по себе очень интересны, так как именно в их реализации разработчик может использовать то, что хочет именно он. Единственное ограничение — это сам разработчик.

В рамках данной работы были реализованы простейшие алгоритмы высокоуровневой логики футболиста. Все они реализованы в классе Player. На данном этапе высокоуровневые методы представляют собой стандартные алгоритмы поведения, в зависимости от окружающей ситуации и позиции на поле. Но при этом отдельно разработаны алгоритмы для полевого игрока и вратаря.

## 4. Перспективы развития

При реализации модуля агента-игрока были использованы простейшие высокоуровневые алгоритмы, для тестирования его функциональности. Была рассмотрена концепция создания простейшего автономного ротоба-футболиста для виртуального мира. В перспективах развития команды по виртуальному футболу предстоит произвести следующие модернизации:

- оптимизация кода низкоуровневой логики;
- добавление дополнительного специфического функционала;
- модернизация высокоуровневой логики:
  - разработка алгоритмов поведения для различных ситуаций;
  - применение подкрепляющего обучения для неизвестных ситуаций;
  - разработка комбинаций для нападения и защиты;
  - применение аппарата нечеткой логики.

### Список использованных источников

1. Noda, I., Matsubara, H.: Soccer server and researches on multi-agent systems. In Kitano, H., ed.: Proceedings of IROS-96 Workshop on RoboCup. (Nov. 1996) 1–7

2. Peter S. Layered Learning in Multi-agent System [D]. Pittsburgh: school of computer

science, Carnegie Mellon University, 1998.

3. M. Riedmiller and Artur Merke, “ Using machine learning techniques in complex multiagent domains,” In I. Stamatescu, W. Menzel, M. Richter and U. Ratsch, editors, Perspectiveson Adaptivity and Learning, LNCS, Springer, 2002.